Optimisation of Data Collection Strategies for Model-Based Evaluation and Decision-Making

In Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

Robert Cain

School of Computing Science

Submitted February, 2016

To Cat, Mum, and Dad

Acknowledgements

I have had the pleasure of working with a lot of people during my varied and unexpectedly long PhD journey. I appreciate everyone who has helped along the way, here I name but a few.

I would first like to thank Aad for his valuable supervision and guidance throughout my PhD. I would like to thank my Mum, Dad, my girlfriend Cat, my brother Chris, and my Grandad for their unwavering moral support and patience. Without the belief of these people in either the work we were doing or my ability to get it done, I likely would not have been able to finish.

Many friends provided guidance and enjoyment during my time at Newcastle University. I can only include a few. Thanks to Derek and Matthew for the early years, Matt, Martin, James, Paul, Becky, and Budi for the later years.

I would also like to thank HP Labs for funding our research 'Prediction and Provenance for Multi-Objective Information Security Management' through its Innovation Research Program, and our original project partners: Doug Eskins, Robin Berthier, Bill Sanders, and Simon Parkin.

Abstract

Probabilistic and stochastic models are routinely used in performance, dependability and, more recently, security evaluation. Models allow predictions to be made about the performance of the current system or alternative configurations. Determining appropriate values for model parameters is a long-standing problem in the practical use of such models. With the increasing emphasis on human aspects and business considerations, data collection to estimate parameter values often gets prohibitively expensive, since it may involve questionnaires, costly audits or additional monitoring and processing. Existing work in this area often simply recommends when more data is needed rather than how much, or allocates additional samples without consideration of the wider data collection problem.

This thesis aims to facilitate the design of optimal data collection strategies for such models, looking especially at applications in security decision-making. The main idea is to model the uncertainty of potential data collection strategies, and determine its influence on output accuracy by using and solving the model. This thesis provides a discussion of the factors affecting the data collection problem and then defines it formally as an optimisation problem. A number of methods for modelling data collection uncertainty are presented and these methods provide the basis for solvable algorithms. An implementation of the algorithms in MATLAB will be explained and then demonstrated using a business workflow model, and other smaller examples. These methods will be presented, tested, and evaluated with a number of efficiency improvements based upon importance sampling and design of experiment techniques.

Collaborations

The research described in this thesis began as part of a collaborative project with the University of Illinois at Urbana-Champaign (UIUC) and sponsored by HP Labs. The general aim was to improve security decision-making models. The project featured two largely independent strands, one on better data collection for models (Newcastle University) and one on better modelling of human behaviour (UIUC). While the project was highly beneficial for the overall modelling process and the discussion of early ideas, the work presented in this thesis is my own unless appropriately referenced and completed with collaboration of my supervisor Aad van Moorsel.

The PRISM based examples in Chapter 6 were facilitated by working with John Mace at Newcastle University. John created the related models and provided some assistance on integrating them.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

When creating and using stochastic models data must be collected to aid model design and populate input parameters. While ideally we would use perfect data for all aspects of the model, in practice this is not possible due to restrictions on money, time, the collection methods available, and other factors. Decisions must be made on how to allocate data collection resources amongst the available options. Without guidance the data collection decisions would be uninformed and may be costly and inefficient.

Given limited resources, one does not wish to allocate much collection effort to aspects of the model that have little contribution to the quality of output. At the same time one often cannot use all of one's collection resources on one aspect, especially with the prospect of a diminishing return on the investment. For all but the simplest of models the best collection strategy will likely be a non-uniform allocation of collection resources. This thesis creates a framework for describing this problem area and looks at trying to create efficient, model-backed methods for finding the optimal collection strategies. To put it differently, we observe that 'garbage in - garbage out' is a common maxim in modelling literature. The work presented attempts to efficiently prevent the 'garbage in' part with the common assumption that for a good model this should lead to better output.

There has been increased interest in using modelling in the area of information security to simulate and quantify the possible effects of policy decisions on both the systems and users involved. These models require data for creation and optimisation. Due to the sensitive nature, and until the benefit is more proven,

the allowable data collection for this purpose can be limited so it is important that it is efficient and cost effective. While this is the primary application domain for the presented work, the techniques can apply to a range of models if the model is treated mostly as a black box.

The problem of data collection for model input parameters can be broken into levels and be looked at from multiple points of view. At the lowest level of populating a single parameter, one needs to specify what data is needed and then compare the different ways of collecting that data. The collection methods and the data produced will have different qualities which need to be taken into account to decide between them. To make predictions one needs to be able to estimate the effect of allocating variable resources (money, time etc.) to these data collection methods.

Expanding upon that one needs to look at the input parameter in relation to the model and its outputs. To compare inputs one should work out how influential the input is on the output. This is done by measuring its sensitivity: specifically in this context, the sensitivity to the parameter uncertainty. One must also consider which model outputs are important and to what level of accuracy.

Looking at the problem from a higher level, one needs to be able to combine these aspects in a way that allows for comparison between different combinations and allocations (known here as a collection strategy). In turn, this allows predictions to be made about those that are optimal.

Wrapping all of these is the need to make the entire process efficient and computationally viable. Model simulation can be very costly with complex models when exploring the input space. The process of deciding which data to collect should not take too long to estimate and the process cost must be significantly less than the cost of using a non-optimal strategy.

1.1 Aims and Objectives

The aim of this work is to develop a rigourous engineering approach to determine data collection strategies, in order to facilitate better model-based decision-

making. A main target user of our work is the Chief Information Security Officer of a company and his team, assuming they use model-based approaches to making information security decisions. The following primary objectives were identified:

1. Study the literature to identify the state of the art in information security decision-making and associated data collection approaches.

2. Formulate the data collection problem of comparing and deciding data sources for model parameters, beginning with the fundamental aspects.

3. Expand the formulation and optimisation problem complexity over multiple versions, including multiple parameters and variants of the optimisation problem formulation.

4. Develop efficient algorithms to solve the data collection optimisation problems.

5. Develop software tools that find optimal data collection strategies by solving the optimisation problem, using integration with the model software.

6. Apply, evaluate, and improve the optimisation tools using (simulated) case studies.

1.2 Approach

Our approach is as follows: we establish a connection between uncertainty in the input data with uncertainty in the output of the model. Data from data sources that reduce the output uncertainty the most will be selected, since they correspond to more accurate results and thus better justified decisions. This approach implies that uncertainty of the inputs need to be modelled. We believe that in many settings it is natural to assume a Normal distribution around a mean, justified by the fact that when sampling from data sources the central limit theorem can be applied to determine uncertainty in the sample mean.

A potential bottleneck of the proposed approach is the time the basic algorithm takes to determine the optimal strategy. We therefore introduce alternative algorithms, which provide complete or approximate solutions. One includes the reuse of model results across different strategies, relying on importance sampling equations to weigh the results appropriately. The application of our importance sampling inspired speed-up is sensitive to various choices. We study this in detail in the experiments with a simple queueing system. We also attempt to provide an iterative approach using related work of Freimer and Schruben [1] and alternative techniques for strategy space searching that do not require us to evaluate every strategy to find an optimal strategy.

1.3 Example Scenario

To help motivate and explain solutions in the thesis we introduce a recurring example scenario. This will feature in various chapters throughout the thesis.

> **Example.** One wishes to model a **supermarket checkout process**. The model would provide estimated queue times, throughput, and productivity with different configurations, such as the amount of tills, reducing or increasing service time, etc. The level of complexity may vary to appropriately illustrate the related section.

1.4 Publications

This thesis includes work which has been previously included in, or is closely related to, peer-reviewed publications which have been written or co-written by the author.

- R. Cain and A. van Moorsel, "Optimization of Data Collection Strategies for Model-Based Evaluation and Decision-Making," presented at the IEEE/IFIP 42nd International Conference on Dependable Systems & Networks (DSN), 2012. [2]

The above paper summarised the ideas that formed a basis for much of the work that went into this thesis and therefore sections in multiple chapters. The formulation and optimisation problems are edited and expanded in Chapter 3. The Central Limit Theorem based input modelling is in Chapter 4 and the solving algorithms make up part of Chapter 5. A similar queue example is used in Chapter 6, but it is covered in much more detail.

1.5 Thesis Outline

This thesis is structured as follows.

Chapter 1 Introduces the problem, aims, and contributions of the thesis. It details the thesis structure and existing publications.

Chapter 2 Reviews background material and related work to sufficiently understand the area and position the work with respect to the field.

Chapter 3 Formally describes the problem area, including a collection of optimisation problems the later method attempts to solve.

Chapter 4 Presents approaches to modelling input parameter uncertainty for analysis using the model.

Chapter 5 Explains the optimisation solving algorithms as well options to speed up the process or reduce the solution space.

Chapter 6 Demonstrates and tests the methods using the MATLAB implementation and multiple examples. A queuing model is used to test all methods and a PRISM workflow model is evaluated using the most appropriate techniques.

Chapter 7 Summarises the conclusions of this work and briefly discusses future directions that could be explored.

1.6 Contributions

This thesis presents a number of key contributions to the area of data collection for models:

1. A background and literature review of data collection for stochastic and probabilistic models and related model analysis topics.
2. A problem formulation for expressing the data collection problem as a collection of optimisation problems.
3. Algorithms to allow the solving of the optimisation problem, based on computing the model output variance under different strategies.
4. Additional optimisations for improving algorithm efficiency using importance sampling and other techniques.
5. A prototype MATLAB tool allowing generic execution of the optimisation solving algorithms and associated functions.
6. A demonstration and evaluation of the optimisation implementation using multiple examples.

CHAPTER 2

Background and Related Work

This chapter provides background information for the thesis and discusses related research. The chapter explains the key concepts in this cross-discipline area of data collection and computer modelling. It uses a review of existing research to position the work of the thesis.

The remainder of the chapter is structured as follows. Section 2.1 introduces uncertainty and related concepts to guide the chapter. Section 2.2 gives an overview of the types of models considered throughout the thesis. Section 2.3 discusses the data and data collection needed for these kinds of models including the issues involved and where this work fits in. Section 2.4 explains sensitivity and sensitivity analysis in relation to models, including existing anlysis methods. Section 2.5 does the same for uncertainty and relates the two concepts. Section 2.6 reviews further related work more specific to this area, those that tackle similar problems to this thesis. Section 2.7 briefly discusses simulation reduction methods. Section 2.8 provides more specialised model examples that this work is targeting. Section 2.9 highlights the software tools used in this research and related models. The chapter is concluded and briefly summarised in Section 2.10

2.1 Introducing Uncertainty

Uncertainty is common concept throughout this chapter and the remainder of this thesis. We discuss application to models that are unintentionally uncertain

(due to incomplete knowledge about the problem) and intentionally uncertain (to represent the scenario). The information and data used to populate these models may be uncertain, and the techniques used to solve these models efficiently can result in uncertain estimates for model outputs. It is therefore important to define upfront what we mean by uncertainty, and which specific uncertainty we will be concentrating on so that it is appropriately considered while reading the chapter.

Uncertainty is a lack of certainty. Uncertainty represents or quantifies how confident we are that some measurement, prediction, or representation is accurate. If there is large uncertainty in a variable, its true value may largely differ from the estimated value but there is still small chance that it is close to it. We do not know for certain since the true value is usually unknown. We lack complete or perfect information about the scenario.

Accuracy is then a related concept. Accuracy is a quality of whether something is correct or close to the true value. In many modelling situations it is difficult to define if something is accurate since the true value is often unknown. It is sometimes possible to estimate the uncertainty within specific probabilities. A value for the number of installed tills in our supermarket example can easily be measured accurately and with certainty.

In this work we are primarily interested in input parameter data uncertainty (*parameter data uncertainty*, or simply parameter uncertainty). Parameter data uncertainty considers how confident we are in the value used to populate an input parameter for a target model. If non-trivially data collection is needed for a parameter, we likely do not know the true value so we cannot measure how accurate the parameter value is but given other aspects of the scenario we can estimate or represent its uncertainty. Later, we will attempt to efficiently reduce this uncertainty.

It should be noted that uncertainty can have different definitions and usage across domains. We attempt to be consistent with our usage here.

2.2 Modelling

The methods proposed in this thesis try to be as generically applicable as possible to system modelling. The primary application is for probabilistic and stochastic models. These are often used to simulate the behaviour of a system for performance and dependability analysis. These models are designed to replicate a real-world process, which could be anything from a single electrical component to multiple machines in a factory or a representation of natural phenomenon like river flooding. Modelling the system allows the user to make predictions about its performance given some inputted state and configuration. The model may also be used to provide predictions of the effect of alternative system configurations, such as whether adding more tills to a simulated supermarket sufficiently reduces customer queue time.

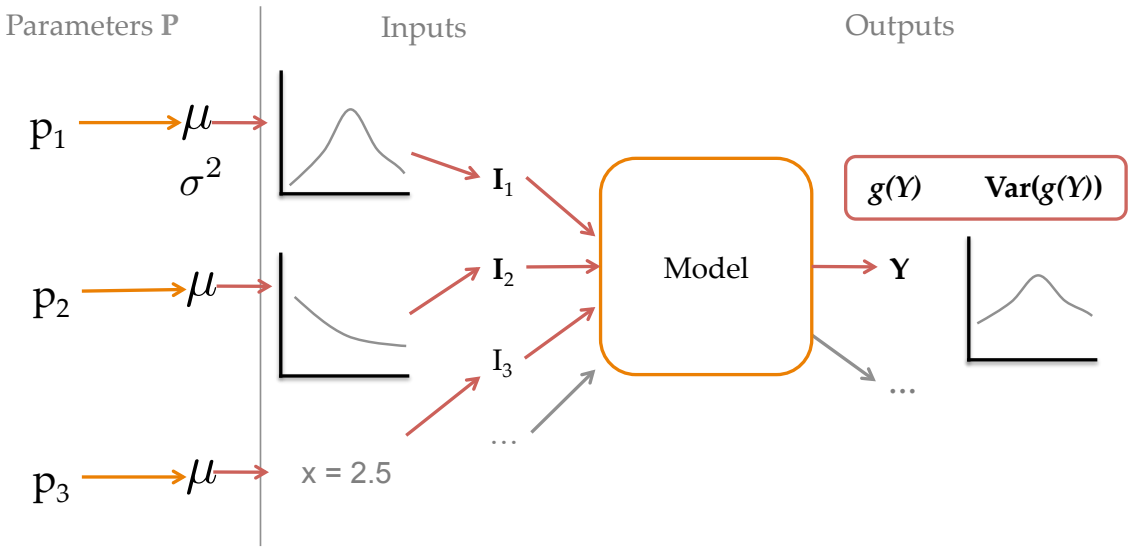


Figure 2.1: Model Parameters, Inputs, and Outputs

Each model takes a variable number of inputs that may be constants or random variables. These inputs are configured to represent their real world values or some new test configuration. The inputs can be categorised in multiple different ways, we use the following (based on [3]):

Experimental inputs (or decision variables) Those that can be changed by the user to test different configurations or treatments e.g. the number of tills open. These may still need an informed starting value.

Model inputs These represent variable inputs in the system that include some randomness but cannot usually be directly controlled e.g. the number of customers arriving at the store.

Fixed inputs Those that are fixed representation of the real system's behaviour. There is no uncertainty during execution and the same value is always used throughout. These may not need data if the true constant value is known.

The model implements an abstract representation of the system process. The level of abstraction will vary based on the complexity of the system, the accuracy requirements, and the available implementation resources. In certain cases a simple linear function combining a small number of inputs can be sufficient in providing a reasonable estimate, others may have multiple interacting components or sub-models and internal feedback [4].

2.2.1 Model Solving

Solving the model calculates one or more output measures based upon the inputs entire probability space. There are two main classifications of model solving techniques: *analytical* and *simulation* [5]. If a model is analytically solvable it is possible to calculate the output's expected value (the value while taking into account the probability of all possible input values) with certainty e.g. $E[Y]$ for output Y. Within *analytical* solutions there are closed-form and analytical/numerical techniques. Using closed-form techniques it is possible to express the output performance measure explicitly in terms of input parameters based upon the model structure. These are often only appropriate for the simpler models. The uncertainty is taken into account and the output performance measure can be calculated directly. Analytic/numerical techniques result in a system of equations, which can be solved using appropriate numerical analysis methods. These methods still provide an exact result.

For simple common model structures and components of larger models, analytical solving is a well covered textbook problem and part of learning modelling theory. It can be difficult and impractical [6] for anything beyond simple models

and simulation techniques are widely used instead.

If a model cannot be analytically solved, an estimate for the output's expected value is produced by repeated execution of the model as part of *simulation* based solving. Simulating the model involves executing for a finite number of steps (or time) to produce one or more outputs. These outputs are measures on the simulated system such as counts of the number of times something happened or average measures taken over the execution. The model user may be interested in both the mean value of the output (or performance measure) and the distribution of the results over repeated simulations or different experimental inputs. The mean is presented together with a measure of its confidence or uncertainty, such as confidence interval or the variance. Replications within the solving allow the resulting estimate to sample from more of the input probability space. Using a larger amount of replications to produce the mean estimate results in lower uncertainty about the value (whether it is near to the expected value). Increasing the number of replications takes more computation time to produce a result so the user or simulation software must balance required confidence in outputs against available time and processing resources.

Improving the effectiveness of these models requires various forms of validation, explained in detail by Sargent [7]. This thesis is primarily concerned with data validity and selection.

2.3 Data Collection

To create the best possible model you need to collect information about the system to be simulated. The modeller needs to understand the process to be modelled to decide the level of abstraction required and then to design and implement it. The data collection process for modelling can be split amongst four stages of modelling (including validation):

Preparation and design: information is needed to correctly represent the process, its inputs, required outputs, interactions, and variability. This goes into informing the design and implementation of the model. It attempts

to realise the appropriate system behaviour but depending upon the complexity of the system it may not produce particularly accurate results at this stage.

Fitting or optimising to the scenario: with a working idea of the system further data may be collected to adapt or tune the model to the system being modelled. This can involve collecting data to help define input parameters and distributions, all the way up to redefining portions of the model.

Validation: further data may need to be collected to formally validate the completed model behaviour, looking at whether the outputs are sufficiently similar to those of the real system, or the agreed expectations of experts.

Ongoing validation and improvement: after its initial usage, ongoing data collection can be used to validate the model's past or current predictions and enable further calibration. If a model was used to predict the effect of a decision, this data collection can be used to evaluate the prediction against the reality and then to improve further predictions in future.

This thesis concentrates on *fitting or optimising to the scenario*, specifically it looks at the inputs and their parameters. It assumes the overall model is sufficiently valid and that improving the data used to define the inputs is the *next best action* for reducing uncertainty in the model's representation and improving the accuracy of model outputs.

In practice it is difficult to define and therefore achieve a sufficiently valid model for complex scenarios [7–9]. An alternative would be to assume that the model is *good enough* but also that its behaviour is being separately and regularly re-validated during the algorithms proposed in the thesis. If a problem is discovered in the model that has a greater effect on the validity than the inputs, it would need to be immediately prioritised over improving inputs. The model solving results and overall result produced by these algorithms may be used as part of this ongoing validation. The model validation technique and its usage are a separate problem not covered by this work.

The methods proposed in this thesis attempt to reduce the uncertainty in the data used to populate the inputs, or the *parameter uncertainty*. If the model's behaviour is correct it should in turn improve the quality of the model outputs. Section 3.2 discusses other data collection aspects of the problem in more detail.

2.4 Sensitivity

Two primary model characteristics affect our optimisation of data collection decisions: sensitivity and uncertainty (Section 2.5). Sensitivity is a measure of the relative response to an action or input. If something is highly sensitive a small change in input will result in a relatively large response. For modelling this is important when identifying the most influential inputs. Inputs which the model is sensitive to have a greater effect on the model outputs and performance measures. On the other hand, insensitive inputs have little to no effect on the model outputs. We therefore wish to prioritise improving the quality of data and allocate additional collection to the more sensitive inputs.

A *sensitivity analysis* is a procedure which assesses the sensitivity of one or more inputs on one or more outputs. It establishes the relative importance of inputs [10]. The results of the analysis can be a simple grouping of inputs [11], relative sensitivity measures [12], or full breakdowns on interactions [13]. The method can consider only the effect of one input singularly, its main effect, or include and review an output's sensitivity to inputs interacting with other inputs. The assessment of secondary and other interactions can help improve the correctness of the sensitivity results. Sensitivity analysis is an active research area in modelling, as it allows us to more easily understand the behaviour of complex models.

While a sensitivity analysis is useful for understanding data collection needs it is primarily important during the model design and validation stages. The modeller needs to understand the model sensitivities to concentrate design and development resources in the right areas. For example, they may wish to abstract away a complex area if the model is actually insensitive to the result [14], or in an

extreme case reduce the model down to a simpler function if the result is dominated by only a small number of inputs. Sensitivity analysis has also been used to optimise model design decisions [15]. Both sensitivity and uncertainty analysis are recommended by international agencies as best practices for validation and auditing [16–18].

Saltelli *et al.* [16] reviews recent work in this area and best practices. A clear purpose is recommended by best practices. The typical purposes of a sensitivity analysis can be split into: *factor prioritisation* and *factor fixing*. Factor and input are often used interchangably. *Factor prioritisation* looks at ordering the inputs or inputs parameters by their sensitivity within the model, the idea being that if the first factor could be fixed at its true value it would reduce the variance in the output the most. *Factor fixing* [19] or screening is the use of sensitivity analysis to find inputs or input parameters which are insensitive that could be fixed at a constant value, or for groups, reduced to simpler abstractions. More formally, the fixing of factor should not greatly reduce the output variance.

Sensitivity can be assessed in many different ways from two perspectives. The analysis may be local or global [20, 21]:

Local Sensitivity Analysis methods measure the sensitivity to deviations from some current input state or values [22]. These methods test sensitivity by using small changes from the current state in one or more input values, and then measuring and comparing the effect on the output. This is easier to compute and is often sufficient, if the current state and local to it are already known to be valid, and if these are the only solution areas under consideration. If not, it may ignore the fact that a model may be more or less sensitive to inputs at majorly different areas of the valid input space. Even with these difficulties this type of analysis still receives a lot of attention [17].

Global Sensitivity Analysis looks at the model sensitivity to an input across its entire valid or expected range, or the entire distribution. The sensitivity is tested systematically at different values. It has the benefit of more complete coverage of the problem space and does not depend upon linearity, monotonicity, or the correctness of a current input value. For models with many inputs, this coverage can

require a lot more model solutions than local analysis, and therefore a lot more simulations. So much so that many methods are often prohibitively expensive when used with complex models [23]. Additional techniques can be used to reduce the number of inputs considered in the sensitivity analysis, or to efficiently and accurately cover the problem space with less model results.

Sensitivity analysis methods can usually be split into two groups: regression-based methods and variance-based methods [21]. The simplest sensitivty analysis techniques are graphical [24], derivative based, or one at a time (OAT). While inappropriate for many models, these kinds of methods are still often used and explored [16]. Derivative methods [25–27] involve executing the model at two different extreme input values, and using difference in output response Y to provide a gradient. These derivatives are compared across inputs or groups of inputs. OAT methods involve testing inputs (or factors) independently, where other inputs are fixed at some predefined value. While unsuitable for many scenarios these techniques can be useful for an easy rough exploration of the model space before inviesting itme in a costly analysis process.

Variance-based methods are especially important to our problem area as they take model-free or black box approach, which makes them applicable to all models and not dependent upon analytic solving. These methods use only the inputs and outputs to understand the model, and compare the variance between model solutions for different input values. Variance-based methods attempt to portion the variance in the output to each input and simulation uncertainty.

The variance separation can be segregated and estimated in different ways depending upon the method. It can be in terms of the main effects [28] (or first order [19]), which estimates the effect of the inputs independently. Alternatively a more complete breakdown of the effects into first order, second order (two inputs together), and beyond, can be produced at higher cost in terms of model runs. The total effects [12] quantifies the effect of an input including its interactions with other inputs as a combined value. It can be produced more efficiently than a many-order breakdown.

Sensitivity analysis methods usually look at the effect of input changes on the

ouput mean, assuming the distribution does not differ significantly. Bar Massada and Carmel [29] propose a method for looking at the effect on the distribution of the ouput using both the mean and variance of the simulation results.

Given the potential complexity concerns involved with running many model simulations, the performance of the methods in relation to the number of inputs and the number of experiments is fundamental. Some of the methods have been compared by Christopher Frey and Patil [24], Ascough *et al.* [30], Chan *et al.* [13] and Hamby [20].

Reducing the inputs to be analysed can be achieved by input or factor screening [31]. The screening filters the input factors to only the most important by removing those that are insensitive. This is usually done with a more basic, much less costly sensitivity analysis method. One that only results in sensitive and insensitive groups, rather than exact measurements.

Kleijnen [31] summarises finding the most important factor by sequential bifurcation, which sequentially tests factors at low and high values while split into groups of decreasing size. The analysis compares results from previous groupings to define the next groups (to efficiently search) and to decide the final important factors. Kleijnen [11] discusses continued work in this area, as it has become more necessary as we make increasingly complex models. Sequential bifurcation requires monotonicity and known signs for the effect of parameters so cannot be applied to all more models. More recently, Shi and Liu [32] extended it to incorporate multiple model responses (outputs), which is beneficial when there is not just one primary output in the data collection problem scenario. Input screening does not need to be entirely independent of later more granulated sensitivity analysis. Campolongo *et al.* [23] propose a unified approach starting with screening and then going on to calculate sensitivity indexes for the identified important parameters. This is a one at a time (OAT) technique, which is not suitable for all scenarios.

The filtering may also be done manually by the user, if they decide one or more inputs can be disregarded as their analysis results would provide no benefit. For data collection, an example would be that if no additional data collection was

possible for an input, it may be filtered out prior to analysis. This kind of user involvement should be avoided where possible, as given extreme results it may become important to re-assess the decision to restrict colleciton.

Efficient coverage of the problem space with limited simulation results is important for both sensitivity analysis and model solving in general, therefore similar techniques can apply. Reduced usage of randomness using a stratified approach to model experiments can result in analysis results that accurately cover the solution space with much less computation [14, 33, 34]. It is important to make sure all extreme regions of the problem space are adequately represented. Both sensitivity and uncertainty analysis methods often use stratified and quasi-random [19] sampling techniques, such latin hyper cube sampling [28, 34], to efficiently create orthongonal experiments and increase input space coverage.

2.5 Uncertainty

Equally important to our problem as sensitivity is uncertainty. As mentioned earlier, in modelling terms we can consider uncertainty in inputs as random variables, input parameters, uncertainty in the design of the model, and uncertainty in the results from solving the model.

These uncertainties are commonly categorised as either: aleatory uncertainty and epistemic uncertainty [35, 36]. Oberkampf *et al.* [37] defines *aleatory uncertainty* as: the inherent variation associated with the physical system or the environment under consideration. It is the uncertainty intentional built into the model, using probability distribution functions to represent the expected variability of the system. Aleatory uncertainty is also called stochastic or irreducible uncertainty [35]. This uncertainty, combined with the structure of the model, is what makes efficient data collection more difficult, more uneven, and in turn the optimisation more important.

Epistemic uncertainty is defined as uncertainty due to a lack of knowledge in any phase of the modelling process [37] e.g. modelling, data collection, validating. For example, the lack of knowledge could be related to the model design

decisions, input data, simulation and solving choices, or validation method. Epistemic uncertainty (or ignorance uncertainty) is the reducible uncertainty that diminishes with improved or additional knowledge of the related factor, which is usually as a result of more effort or complexity in that area. We concentrate on the parameter knowledge part of the epistemic uncertainty.

From the lowest level, data must be collected for a single parameter. A measurement will be taken producing a value for the parameter. The measuring process will often only be certain within some set of error bounds and the value may vary depending on how and when the measurement is taken. Repeated measurements or samples are used to increase the certainty that measurements are accurate. It also ensures that the results as a whole take into account and represent the values real variability.

When these measurements are used in modelling context, variation in the values between usage or over time may be modelled as a random variable. This random variable allows the value produced and used as an input within the model to vary based on some specified probabilities. Random variables have a probability distribution and one or more parameters. The value produced is intentionally uncertain [38] to represent the recorded behaviour but has an expected value if all probable values are taken into account. External to this there can be uncertainty in the correctness of the parameters and the choice of distribution. These are often referred to as parameter uncertainty and model uncertainty respectively, and collectively as input model uncertainty [38, 39]. The thesis covers only reducing the parameter uncertainty. The assumption is that the other uncertainties have already been minimised while producing a sufficiently valid model. While uncertainty over distribution selection is discussed, this is outside the scope of the thesis. It has been covered in other work [40–43] and can be a sufficiently complex problem alone.

At a higher level a model, and in turn its outputs, have multiple causes of uncertainty. The stochastic nature of the models we consider allow the usage of random variables as input, which enables randomness in the simulation process. This stochastic uncertainty [39] means that for the same input parameter values,

the model output can differ between repeated executions (unless repeated with the same random seed). Model solving by simulation is affected by this uncertainty, solving must take it into account to produce a confident result. Results are often reported with their respective confidence intervals but do not usually take into account the uncertainty of the chosen parameter values [44].

The design and implementation of the model as whole can be uncertain. This is largely an extension of the problem of uncertainty in the choice of input distributions and the validation that the model is sufficiently valid. These two model-level uncertainties are sometimes separated as model error (measured in terms of bias) and simulation error (measuring in terms of variance) [45].

The term *uncertainty analysis* defines a general class of methods that try measure the effect of model uncertainty (especially inputs) on model outputs. Input uncertainty analysis methods can be a split into two categories: methods that try to improve output confidence intervals to better take into account input uncertainty [6, 46, 47] (sometimes described as looking at the risk involved in the model or parameters being uncertain), and methods that try to assess the effect of parameter or distribution uncertainty on the model [1, 48, 49] (recommending areas to improve first or the effect of increased certainty). The latter can overlap with sensitivity analysis and the two are often completed together. The work of this thesis can be considered as a form of input uncertainty analysis: analysing the effect of parameter uncertainty and model sensitivity to provide recommendations on the optimal data collection strategy to reduce it.

Kleijnen [48] is one of the first to suggest using confidence intervals and the Central Limit Theorem to represent parameter uncertainty in analysis. Specifying that this could be used as either sensitivity analysis (quantifying the effect): as part of deciding where would benefit from more data, or uncertainty analysis (as risk analysis): trying to quantify the risks the parameter uncertainty has on outputs. The approach does not continue much into the effect of additional collection or consider the wider collection problem beyond that sometimes further collection is not possible.

Barton [6] surveys input uncertainty issues, early solutions and recent progress

in the modelling simulation context. The methods look at the 'propagation of input model uncertainty to output uncertainty' for a mixture of parameter and distribution uncertainty. Four main current approaches are identified: direct bootstrap resampling (including direct resampling), Bayesian model averaging, delta method, and metamodel-assisted boostrapping [46]. Apart from the delta method, these uncertainty analysis methods attempt to calculate new model output confidence interval, which take into account parameter uncertainty or model uncertainty.

Using bootstrap resampling is common in many uncertainty analysis methods. It was first described in [50] to account for the uncertainty in input distribution correctness within output confidence intervals [51, 52]. We describe the process and our usage in Section 4.6. More recently, Barton and Nelson [46] combine bootstrap resampling with stochastic kriging metamodelling [53] to reduce the simulations required.

Ankenman and Nelson [54] proposes a 'quick' bootstrap-based method for assessing the overall effect of input uncertainty from fitting distributions (or parameters) when based on real world data, relative to simulation uncertainty. Since it was not in fact quick, Song and Nelson [55] demonstrate efficiency improvements. The method provides estimates for the sensitivity to additional data collection. The user would then need to decide and allocated collection based on these values.

Pappenberger and Beven [56] discuss some of the issues with using uncertainty analysis in practice. In general, the main issues revolve around communicating what the results mean to the relevant party. Effectively, it extends the already difficult problem of explaining the uncertainities of complex models to parties without modelling backgrounds.

Like sensitivity analysis, uncertainty analysis can be costly in terms model runs. Instead of the usual experiment reduction approach, Liu *et al.* [57] describe and test an implementation that uses cloud computing resources, specifically Windows Azure, to speed up achieving a result for groundwater flow models. This needs to account for the issues with the portability of model software and

licensing of MATLAB for cloud usage. The licensing issue should now be less of an issue as MathWorks (makers of MATLAB) work towards their own cloud integrated solution [58].

Uncertainty is a problem throughout model usage from data collection via model solving and into the produced outputs [59]. Methods must be used to minimise this uncertainty, as much as is practical, to a level which is appropriate for the problem. The thesis provides an extended discussion of this aspect of modelling, a problem formulation of a chosen part of the uncertainty problem, and multiple approaches attempting to provide solutions that reduce parameter uncertainty while taking into account other aspects of uncertainty where appropriate.

2.6 Optimising Data Collection

The thesis aims to allocate data collection resources based on the uncertainty of data, the needs of model, and within the limits of the collection scenario. The closest existing methods for allocating more data look only at the first two or simply highlight the need for more data. Freimer and Schruben [1] presented a similar approach to those proposed in the thesis and provides a basis for certain methods discussed later. While they provide a way to allocate additional samples based upon the model, the method includes no data collection factors beyond the existing samples and the number of samples. Samples are allocated to an input rather than any consideration of where the data is coming from with other related factors such as costs. Freimer and Schruben [1] highlights taking into account collection cost as important future work for methods allocating additional samples.

Ng and Chick [49, 60] explain a Bayesian based approach [61] to reducing parameter uncertainty by optimising the allocation of samples. It relies upon creating an output response function in term of the input parameters to use in place of the model. They briefly propose representing the process as an optimisation problem but with limited detail. The model is replaced with a response surface metamodel using Bayesian model averaging [62], this kind of response surface

methodology is not only useful for the optimisation approach but also used more widely in model analysis when direct model execution is too costly. This includes both sensitivity analysis [63] and uncertainty analysis [60] methods.

Many data collection qualities are difficult to measure quantitively, solutions to their assessment are usually subjective, which can lead to similar optimisation. An expert opinion or the user giving qualities a score can be transformed into a combined total but cannot be used with the model. Model sensitivity needs to be assessed separately. This kind of assessment is more useful for choosing one data source for an input rather than complete optimisation.

Närman *et al.* [64] tackles the problem of cost effective data collection but for the purpose of software quality analysis. The method uses diagnosis in Bayesian networks to choose which data sources are likely provide the most value for analysis with the least cost. The evaluation of data source value and cost is done in subjective manner, which this work will attempt to avoid as much as possible, in favour of measurable aspects.

Skoogh *et al.* [65] take an alternative approach to optimising data collection by integrating ongoing data collection into the modelling process. In essence, they are optimising the time it takes to get an updated model based on new data. This kind of data collection could be used for some parameters in the models we target but it needs to be customised based on the scenario and modelling tools used.

2.7 Simulation Reduction

Solving by simulation can be costly for complex models containing a lot of inputs and stochastic uncertainty. When solving by simulation the user wishes to reduce the variance (uncertainty) due to simulation in the result, while also limiting the number of runs or simulation time used in simulation. We can try to reduce the cost of simulation by variance reduction methods (improving the efficiency of the simulation) or replacing the model with a response surface or metamodel (using simulation less).

Variance reduction is a strong area of research in modelling, since the success

of these kinds of models depends upon their usage being computationally viable. Techniques of interest here include: input filtering, sample design, design of experiments, and importance sampling. Input filtering or screening has been mentioned previously in terms of sensitivity analysis but the fixing of an unimportant uncertain input can also contribute to simulation variance reduction. Instead of completely random sampling from input distributions, one can increase the rate of convergence using sample designs and design of experiments. Sample designs including stratified sampling, ensure coverage of the input space. Design of experiments [66], such as orthogonal arrays or latin hypercube sampling, ensure coverage of the input space collectively. Importance sampling allows the re-weighting of results based on the difference in probability distributions [67, 68].

Regression techniques allow us to replace a complex model with a simpler, less costly model. For stochastic models, this kind of replacement is referred to as response surface methodology [69] and metamodels. These approaches still require some simulation, as the inputs and outputs of the original model are used to select an appropriate metamodel and then train or optimising the meta models results to be an adequate representation of the original model. Response surfaces are not only used for simulation reduction, sometimes their usage is simply to pre-simulate all the required solution space.

2.8 Security decision-making Models

Computer security investment decision-making is largely based on Chief Information Security Officer (CISO) experience and expert opinion. This knowledge may provide many good decisions but provides limited auditable evidence, essential for supporting big investments in a business environment. One possible alternative is using modelling to: assist in a security investment decision making; add evidence to these decisions; and highlight potentially unforeseen consequences of these decisions.

Using models for security decision-making is becoming more popular [70], es-

pecially in the area of information security [71–75]. The purpose of these models is to simulate a computer system, business organisation, or attacker and predict how specific security policy decisions may affect the system outputs. The security policy will dictate what security controls are implemented, which in turns affects user behaviour and results. Two examples of this are password usage requirements [72] and the usage of USB flash drives [71]. The security policy can enforce or recommend the use of certain levels of password complexity or frequency of password changes. This policy and the workers related behaviour will have an effect on the business that we may wish to estimate prior to, or during, its implementation.

The modelling of information security can cross over into economics style modelling [76], trying to estimate the utility for the organisation, especially in terms of trade-offs and costs. A common trade-off in these models and their outputs is Security vs. Productivity. The aim of the policy choice is to balance the security of the business' data against the potential cost of reduced productivity or more limited availability. A business does not want to invest in a highly secure system that no one can use, which prevents work being done and money being made. They also usually do not want to be completely open to attack just to make everything easy for the employees.

Due the complex mathematical and uncertain nature of this kind of modelling, care must be taken when presenting and communicating these results to interested users. Parkin *et al.* [77] considers designing interfaces for CISO decision making tools, including controlling the experimental parameters and displaying the results from a completed model.

2.9 Software Tools

In this section we highlight third party software and libraries that were used in the development of research solutions.

There are many software tools available to enable system modelling and analysis. We only discuss those used in this thesis, which were either already in use

or the most readily available. The primary software implementation of the research was completed in MathWorks MATLAB (R2013a to R2014b) [78]. MATLAB is both a programming language and development environment for numerical computing. MATLAB has many different toolboxes for additional features crossing multiple domains, including Simulink and SimEvents [79] for creating and simulating different types of models.

PRISM [80, 81] is a Probabilistic Symbolic Model checker, it is a formal modelling tool that allows us to create and analyse many different types of probabilistic models. The tool assists in writing models in the PRISM modelling language and it allows for model solving using both verification and simulation. It will be used as part of the examples in Chapter 6.

2.10 Summary

This chapter introduced and explained the key background topics for this problem area. This background knowledge allows us to have sufficient understanding of the problem domain leading into subsequent chapters. Part of this included reviewing relevant research from this area in more detail. Doing so highlighted gaps and areas for improvement in the current research, we hope to push forward some of these with this thesis.

The next chapter uses this knowledge to present a formal framework for our data collection problem. The problem of optimising data collection for models is considered from multiple approaches and at different levels of complexity. The most relevant related work [1] is used as a basis for Section 3.3.2.3.

CHAPTER 3

Defining Data Collection as Optimisation Problems

The previous chapter provided an overview of the cross-discipline problem area and a review of existing research into similar problems. The following chapter provides a formal description or framework for describing our problem domain. This will be continually used and built upon throughout the remainder of the thesis. We discuss key data and data collection attributes and metrics before defining the problem as an optimisation problem to be solved in later chapters.

Section 3.2 breaks down and explains important variable aspects of data and data collection that were considered for the problem formulation. Section 3.3 formally defines the problem of collecting data for model parameters and optimising strategies. This starts from fundamental variables of the problem, building into more complex optimisation problems from multiple different approaches. In Section 3.4 we summarise the content and key contributions of the chapter.

3.1 Introduction

To evaluate and recommend data collection strategies that will improve model output quality we must further explore the details and scope of the problem. This will involve breaking down the problem and looking at variable aspects that can be controlled and then optimised. We need to define what qualities are im-

portant to measure and how these could be related to an overall strategy quality to optimise.

> **Example.** Consider a complex version of our supermarket model. There are multiple different parameters, e.g. average time between customers arriving, probability of purchasing, average time to serve customer, difference in service time between till operators, probability of till failure, number of tills available, and number of employees available etc. There are multiple ways of getting data for each with different properties. There exists restrictions on the allowed collection. We want to select a data collection method for each parameter and allocate certain amount of collection resources depending upon the potential improvement provided.

3.1.1 Research Questions

- How can we quantitatively compare data sources and data, specifically for model usage?
- How can we relatively quantify the importance of model parameters?
- How can we use these in combination to evaluate data sources across different parameters and in turn complete strategies?

These can also be inverted, looking at it from the defining an optimisation perspective:

- What defines a good data collection strategy?
- What is the objective or objectives? The goals of optimising the data collection.
- What are the problem constraints? The restrictions that will be put upon the solution space that defines allowable collection strategies.

3.1.2 Approach

The optimisation process (Figure 3.1) must take into account the model's input parameters, the available data collection methods, and the stakeholder or model user's requirements. The input parameters have requirements, a current state, and different levels of importance to the model. The data collection methods have various attributes that provide data of differing quality based on the type of collection and the amount of investment. The stakeholder will specify restrictions on the optimisation process, such as the allowed collection methods, required quality levels, and available collection budget.

There are many aspects of data and data collection quality that could be used but some are not essential to the fundamental problem. We will review different quality metrics, define those that continue into later chapters, and those that could be added in future work. Starting from a simplified scenario we will build optimisation problems of increasing complexity. These need to sufficiently represent the most important aspects of the problem area for prototype solutions to be implemented in subsequent chapters.

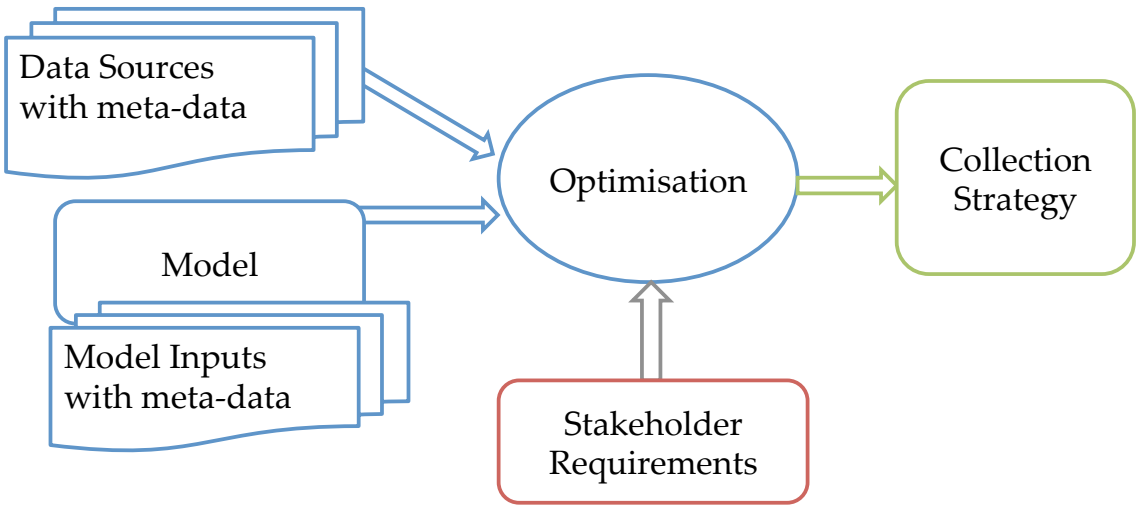


Figure 3.1: Optimisation Problem Overview

3.2 Data and Data Collection Quality

This section provides an extended explanation of data collection methods modelled as data sources. We first define data and data sources before reviewing important data and data source attributes. There is a short discussion of data quality measures useful for the problem area that were considered for the formulation. This is followed by a summary of how these fit into the problem formulation in Section 3.3.

3.2.1 Data and Data Sources

The term data is used here for information collected to accurately represent the system being modelled. This data can be collected by a variety methods including direct measurement and monitoring, questionnaires, and expert opinions. The method of collection and the data itself have attributes that need to be measured and considered, when choosing how to collect the necessary data and how much is needed.

To collect data for a model parameter you take a measurement of the related system attribute either directly via instrumentation, or indirectly e.g. from the users via a questionnaire, either results in one value. To make sure this information is correct multiple additional values are usually collected (known as samples). These samples can then be evaluated using common statistical techniques to inspect and analyse the behaviour of the results, and decide upon an estimate to use for the input parameter.

In the following formulation, data collection methods are abstractly represented as a sampling based *data source*. Our problem considers using data collection results from a limited to set period of time. We do not discuss using or integrating streams of data although some ideas could apply. Non-sampling based sources (such as an expert opinion) would need to be transformed into a similar representation. This is done to allow comparison of measurable attributes in the proposed methods. Quality assessment of subjective data is outside the scope of this thesis. We assume data from a data source is separately validated for model

usage.

Uncertainty analysis allows us to consider which parameters are most in need of additional data, by analysing the effect of their uncertainty on the model output. This allows one to compare the effect of parameter data uncertainty across inputs, taking into account their importance in the model. Separate to that, and at a per parameter level, there may be multiple data sources available. The attributes of the data source, and the data it produces, need to be comparable and considered appropriately. In the next sections those we consider most important to modelling and the thesis are discussed.

3.2.2 Data Source Variables

This section discusses important data collection attributes to consider when collecting data for models and in general. Some of these are then translated into variables that become part of the problem formulation in Section 3.3.

Data collection attributes related to a collection method or source:

3.2.2.1 Sample size

How many samples are needed to provide a sufficiently accurate estimate for the parameter? How can samples be allocated? The number of samples is fundamentally important as the primary controllable aspect (beyond choosing which data source). The number of samples needed will vary depending upon the quality/variability of the data and the importance of the parameter. To take this into account a pilot collection may be carried out or information may be inferred from historical data from the source or from the kind collection method. The units of a sample are left undefined since it can be dependent upon the data source.

For each data source there will be requirements about how samples can be allocated to the source. These requirements result in a set of valid values of the sample size. For new collection with a data source there exists a minimum number of samples, a maximum number of samples, and a distribution that represents valid sample increments in between. For practical purposes the minimum

number of samples is both how many samples are produced from the minimum investment in a collection method and the amount of samples required to produce a value for the input parameter. From a data collection perspective only, the simplest case for a minimum is one sample. In real examples, most data sources will produce multiple samples for the minimum to offset the initial cost of setting up collection (discussed further in Section 3.2.2.3).

The maximum number of samples represents the upper limit on the allowed collection with a data source. This can be as a result of stakeholder restrictions on collection or practical limits on plausible collection. For example, if you are collecting data from a population of users within an organisation and each user can provide only one usable data point, then the maximum allowable number of samples is equal to the population size. There can also be physical limits on the maximum number of samples which could be collected in the timespan available for collection. The stakeholder can also put restrictions on the data collection which could affect the maximum sample size for certain data sources. Returning to the previous example, the stakeholder might limit the amount of users that are made available to the data collection. Any stakeholder limitations on access to the organisation or system being modelled will likely affect the maximum sample size for some of the data sources, for the rest the available budget will likely stop further collection rather than per data source maximums.

How the sample size can be incremented with additional investment can also vary between data sources. In the most basic case single samples can be purchased, the collection sample size can then be any integer between the minimum and maximum sample size for that particular data source. In some situations samples will be allocated in batches, making the sample increment a number greater than one. A method that collects a data sample once per hour for an entire day, may be controllable by the number of days. This results in a sample size increment of 24 (and a minimum of at least 24). In these simple cases, the sample size increments uniformly but it does not need to be. The sample increment can be a function that depends upon the current sample size. For example, a data source may allow only large sample size incrementing initially and for sample sizes less than a certain value, but beyond that value small sample size

incrementing may be allowed as it becomes more cost effective.

3.2.2.2 Frequency and Timeframe

How often is the valued measured and over what length time? Collecting only over a small timeframe or infrequently may result in aspects of the system behaviour being missed or anomalies over-emphasised. More frequent measuring and measuring over the entire system life cycle are likely to achieve the best results but in turn will create higher costs. These time attributes need to be balanced and taken into account by the data collector, who must choose how best to distribute allocations. This is highly dependent on the external collection method and input to be populated. It is assumed that additional resources allocated to a data source are used in the best possible and most appropriate sampling design. The effect of these attributes can be encapsulated in the resulting quality and cost.

3.2.2.3 Cost of collection, additional samples and indirect costs

What does it cost to produce a set of samples and how does this cost change with differing sample sizes? Each data collection method can have different associated costs to collect and process the data. Collection may have a variety of costs including collectors time, equipment and other resources needed to complete the collection. This can sometimes include a cost on the system being collected from e.g. productivity loss. These costs are all considered as a combined into one monetary cost for collection.

In the simplest case, all collecting has a uniform cost, the marginal cost of one sample costs the same as the next. For more complex techniques, such as those requiring major upfront outlays for setup or equipment, the first samples will have a large cost then subsequent additional samples can have a much lower cost. Some data sources will follow an 'economies of scale' -like [82] behaviour, where the average (sample) cost decreases as the quantity increases, from spreading the initial outlay over more samples or reducing collection costs with size. It can also follow the similar limit whereby average cost begins to increase after a certain point.

The non-uniformity and indirect costs must be taken into account when comparing data sources based on total cost. If significant data collection has a negative affect on the system being modelled, the real cost of additional samples may also increase as collection becomes a burden on the system. For example, software monitoring and measuring the system too frequently may consume excessive computer resources and result in less or no productivity. Alternatively, questioning all system users often will result in no less actual work-time. The proposed optimal strategies should be realistic in their expectations or should provide alternatives.

Collection cost relates to all of the previous attributes. While the per sample cost can vary in either direction, the total collection cost for a data source will increase as sample size increases. Increasing the frequency of data collection will increase collection cost, either directly from the collection process or indirectly from the effect upon the system. The same applies for increasing the timeframe of the data collection.

3.2.2.4 Collection Type

How is the data collected? Is it from the system itself or from observations? Data can be collected directly from the system by singular measurements or ongoing monitoring. This is not always practical for all inputs but other data may be available via secondary sources such as questioning the users. The results and values produced using questionnaires may not be as accurate, and may vary more than direct measurement but can sometimes cover a longer time frame.

3.2.2.5 Formulation Representation

We now consider how these will be represented in the formulation. A collection strategy s is a tuple $\{D, N(D), T(s)\}$, which is made up of a set of available data sources D, a set of sample sizes $N(D)$ unique to this strategy configuration, and a total cost $T(s)$. Within D, each data collection method and its resulting data is represented as an abstract data source D_j which has a set of related variables and functions:

Number of samples to collect $N(D_j)$ A quantity of samples to collect at a later date as defined in the data collection strategy. This is the primary attribute that can be controlled as part of the strategy. The others are pre-defined or based upon this quantity.

Sample cost c_j The cost of collecting each sample in future planned collection. This can be a number for uniform sample cost or a function for more complex costing. It encapsulates all previously mentioned cost factors at a per sample level.

Startup cost v_j The initial outlay required to collect data with the data source. This could also be considered as an additional cost on first sample collected for data source.

Minimum sample size, maximum sample size, and Sample increment $N_{min}(D_j), N_{max}(D_j), \eta(D_j)$ The number of samples required as part of new data collection, and a number of samples that must be added at each level of investment in a data source. We restrict the scope of the current problem to only constant increments.

Data source total cost $C_j(N(D_j))$ The combined cost of collecting a chosen amount of new data samples. The total cost of a data source is a function related to the number of samples needed $N(D_j)$. The function's structure will be vary depending upon the complexity of the source. This encapsulates all previously mentioned cost factors at a per collection level. While more advanced cases are discussed, the data sources in the implemented examples in Chapter 6 use only linear functions with an optional constant (startup cost) for their cost functions.

3.2.3 Data Quality

To differentiate data sources one must consider the qualities of any already collected existing data and those expected from future data collection. While we generally refer to the concept of parameter data certainty, there are many data quality dimensions that can be used to define what makes good data. These can

be related to what makes better or more certain data for model parameters. Batini and Scannapieca [83] defines the following dimensions of data quality. While largely orientated for data in terms of records, similar dimensions apply to data sources in the modelling scenario:

Accuracy: given a real world value and an estimate, accuracy measures the closeness of the estimate to the real value. More generically, this can be split into accuracy in terms of whether value is valid and appropriate, and whether or not it is correct. Considering quantitative data collected for parameters, we are interested in how well the mean represents the population mean. One can also look at it from the other perspective, when not directly measured, one must consider whether the target or environment used for the data collection is an accurate representation of the system being modelled.

Completeness: This quality is more appropriate for data as records where an entry is made up of multiple values [83]. When using multiple samples from a system, one can also consider completeness in terms of whether these samples completely represent the system. For example, if collecting data for mean inter-arrival time of customers, the data may be an incomplete representation if only collected for one hour on a Friday afternoon. Similar applies from a users perspective, the mean parameter designed to represent all users should not be populated based on data from only one type of user. This partly overlaps with the previous dimension as it can be considered as a specific cause of inaccuracy.

Time dimensions: Currency, Volatility: how recently was the data created or collected and how does the data qualities change with time. Since we are optimising future collection currency is less of an issue but it may still have an effect. For example, there may be a data collection choice between an expensive data source that can provide highly current data, and a data source which is cheaper due to the data being less current. The volatility dictates whether this currency is likely to make a difference. The currency and volatility are also important when basing data collection decisions on past data.

Consistency: This dimension looks at the integrity of data over multiple entries, or samples in this context. One can consider the consistency of measurements values or the consistency of data qualities during a single collection. While accuracy looks at the distance from the real value, precision describe how consistently close repeated measures are to the same value.

Accessibility: How easily can one access and integrate the required data? The data may require additional transformation to be used. While mostly affecting the data source cost, this may affect data qualities based on the transformations required especially if it involves combining with other data.

There are many ways of looking at the quality of groups of data and the importance of each can vary depending on the situation [84]. Most are suitable for consideration over sets of samples for a target value. Many of the qualities are difficult to measure, and often highly subjective. It is essential to the optimisation that attributes be comparable across inputs, or at least able to be representable that way. In this context of sampling-based data for parameters, we are primarily interested in the more low level accuracy and variability of the data collected. For this one can look at the variance of data from the data source. This provides a measure of the variability of the data at the current sample size. As one will see in Chapter 4 this can be used to represent the parameter data uncertainty.

The following variables represent the data quality of a data source in the formulation:

Existing samples Each source has a small set of existing samples $\{x_1, ..., x_{M(D_j)}\}$, or a least a $\hat{\mu}_j$ sample mean, $\hat{\sigma}_j^2$ variance, and $M(D_j)$ previous sample size. We assume there is some existing data from prior collection during building the model creation, a pilot study, or inferred from historical data for the collection method. This assumption allows us to represent the current and predicted data uncertainty in Chapter 4. Estimating this information, if unavailable, is outside the scope of this thesis. If the data source can collect data for more than one input parameter (see Section 3.3.2.2) there must be existing data for each parameter.

One could encapsulate other attributes with multipliers but these would be very difficult to quantify, adding an extra layer of uncertainty. Similar is true for translating expert opinion (and certain aspects of qualitative sources) into a sampling-based representation, which are outside of the scope of this thesis.

3.3 Problem Formulation

This section formalises our problem in Section 3.3.1, phrasing the problem of finding the best data collection strategy as a mathematical programming problem in Section 3.3.2.

3.3.1 Formal Problem Definition

The models we consider are discrete-event dynamic systems [85], of any type, be it Markov chains, stochastic process algebras or discrete event simulation code, and may also include rewards (Markov reward models etc.). The model M takes as input a set P of *input parameters* or *inputs*: $P = \{p_1, \ldots, p_{|P|}\}$. The output of the model is a random variable Y, which is a function of M and P and typically one would be interested in a function $g(Y)$ for instance the mean of Y or some other reward function over Y. In a queueing system, Y may be the steady-state queue length distribution and we could be interested in the mean number in the queue, the holding cost of jobs, etc.

Without loss of generality, we restrict the set P to the input parameters for which data can be collected, and we consider only 'individual parameters' (not full distributions), such as the mean or variance of a distribution or a probability used in M. For each input parameter $p_i \in P$ there may be multiple data sources available. The set D_i contains all available *data sources* for input parameter $p_i \in P$, and with $|D_i|$ the number of data sources, we write $D_i = \{D_{i,1}, \ldots, D_{i,|D_i|}\}$. A data source may be a set of samples from a sensor or the result of a set of questionnaires, etc. We write $D = \cup_{i=1}^{|P|} D_i$ for the set of all available data sources.

A *data collection strategy* s in the set S of all strategies is a subset of all possible

data sources D. In its most general form $S \subset \cup_{i=1}^{|P|} \cup_{j=1}^{|D_i|} D_{i,j}$, but if appropriate we can restrict the set of possible strategies. For instance, we may assume that valid strategies select for each input parameter $p_i \in P$ one and only one data source from the set D_i. Let the chosen data source be denoted d_i, then we can write the strategy as $s = \{d_1, \ldots, d_{|P|}\}$. If appropriate, we use $|s|$ to denote the number of data sources in the strategy $s \in S$, so in this special case $|s| = |P|$. Another special case is that we want to optimise the number of samples from a data source; since that is the common case in this thesis, it is useful to introduce bespoke notation for this case (instead of assuming each different sample count is a different data source). The *sample count* for data source $D_{i,j} \in D$ is denoted as $N(D_{i,j})$ and a strategy s is denoted as $s = \{N(D_{i,j})\}, i = 1, \ldots, |P|, j = 1, \ldots, |D_i|$.

An example best illustrates the approach to formulating the optimisation problem.

> **Example.** Assume our supermarket model needs the mean number of people arriving in the store. Consider two different sources for this single input parameter: (i) a questionnaire of a subset of the population and (ii) live counting at the store entrance for one hour each day for a number of weeks. We do not expect the two sources to give a different result (they are both unbiased, and with enough samples would converge to the same value for the model input parameter). However, neglecting the cost of each approach, we may expect that counting at the store entrance gives more accurate results than questioning arbitrary people, and thus will lead to more accurate output results.

To represent this thinking in an optimisation problem, we need expressions for the uncertainty in input parameters caused by the various data sources. We also need to identify for each possible strategy how this uncertainty propagates to uncertainty in the output $g(Y)|s$.

For a strategy $s \in S$, we model the uncertainty in the outcome of a data source by a random variable $X_{i,j}(s)$ for parameter $p_i \in P$ and data source $D_{i,j}$, thus resulting in the multi-dimensional random variable:

$$X(s) = \{X_{i,j}(s)\}. \tag{3.1}$$

The uncertainty about the output random variable then depends on $X(s)$, and we can introduce a random variable $g(Y)|X(s)$, which reflects in the output the uncertainty of input sources. To attain an effective optimisation criterion, we use the variance of $g(Y)|X(s)$, which we refer to as the *output variance* (for the strategy). It should be noted that alternative metrics can be used (such as correlation ratios [13]), but the output's variance is a common metric [1, 14] and suffices for our purposes. The specific expression for the output variance $Var[g(Y)|X(s)]$ depends on the result we want out of the model. For instance, if we are interested in $E[Y]$, then the variance would be $Var[E[Y]|X(s)]$. In what follows we usually drop X from the notation and write $Var[g(Y)|s]$ to denote the uncertainty in output under the data collection strategy s.

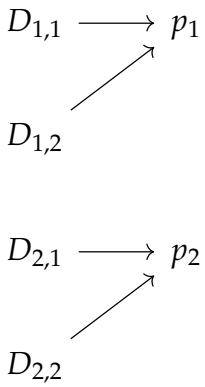
It is important to comment on the subtle difference between $Var[Y]$ and $Var[E[Y]|X(s)]$. $Var[Y]$ is a metric of interest in its own right [29], which can be computed directly from the model, as for instance in [86]. $Var[Y]$ has no relation with any data collection strategy s. For our supermarket example $Var[Y]$ is the variance in queue length, for instance. However, in $Var[E[Y]|X(s)]$, the metric of interest is $E[Y]$ (the mean queue length), and $Var[E[Y]|X(s)]$ expresses the uncertainty in the result for $E[Y]$ *caused by the uncertainty in the input parameter introduced by the data collection strategy*. Note that as a consequence, if $X(s)$ has no randomness, $Var[E[Y]|X(s)] = 0$ because there is no uncertainty in the input parameter values.

3.3.2 Mathematical Programming Definition

The previous formal definition of the data collection problem provides us with an objective function for optimal data collection strategies, namely to minimise the output variance $Var[g(Y)|s]$. For practical applications it makes sense to add additional constraints to the optimisation problem, for instance considering the cost of collecting data, or a limit on the total number of samples for each source, etc. One can also swap objective and constraints, thus minimising cost under some

constraint on the output variance. This leads to the following plausible versions of the mathematical programming formulation of the optimisation problem: (1) Minimise output variance given a total number of samples, (2) Minimise output variance within a cost budget, and (3) Minimise the cost for a target output variance. We introduce these first before expanding problem complexity further.

3.3.2.1 Single Parameter Data Sources

> **Example.** Consider a simple version of supermarket model with one till and two parameters. p_1 mean arrival time, p_2 mean service time. If each parameter has two data sources:
>
> 
>

Let $s \in S$ be any possible strategy (that is, $S \subset \cup_{i=1}^{|P|} \cup_{j=1}^{|D_i|} D_{i,j}$), and let $d_{i,j} = 1$ if source $D_{i,j} \in s$, and $d_{i,j} = 0$ otherwise. We first provide the mathematical programming formulation assuming exactly one source will be selected per input (that is, $|s| = |P|$). Importantly, a strategy is variable in terms of the number of samples associated with each chosen source, so a strategy $s \in S$ is determined by the number of samples $N(D_{i,j})$. Assume now that the total number of samples has an upper bound N, possibly because the time to collect is limited. The following optimisation formulation is then natural:

Optimisation Problem 1 (Sample Constraint)**.**

$$Min_{s \in S} \;\; Var[g(Y)|s]$$

subject to:

$d_{i,j} \in \{0,1\}$ *for* $i = 1,\ldots,|P|, j = 1,\ldots,|D_i|$

$\sum_{j=1}^{|D_i|} d_{i,j} = 1$ *for* $i = 1,\ldots,|P|$

$\sum_{i=1}^{|P|} \sum_{j=1}^{|D_i|} d_{i,j} \times N(D_{i,j}) \leq N$

It is natural to enhance the above optimisation problem with a budgeting constraint C. This allows one to consider the cost of sampling, or the relative effort spent on different data sources. To illustrate the formalisation, assume in above Sample Constraint problem formulation that the cost of a sample of source $D_{i,j}$ is given as $c_{i,j}$ and initially there are no other costs. The budget constraint then limits the valid strategies as follows:

Optimisation Problem 2A (Budget Constraint)**.**

$Min_{s \in S}\ \ Var[g(Y)|s]$

subject to:

$d_{i,j} \in \{0,1\}$ *for* $i = 1,\ldots,|P|, j = 1,\ldots,|D_i|$

$\sum_{j=1}^{|D_i|} d_{i,j} = 1$ *for* $i = 1,\ldots,|P|$

$\sum_{i=1}^{|P|} \sum_{j=1}^{|D_i|} d_{i,j} \times N(D_{i,j} \times c_{i,j}) \leq C$

This can easily be expanded with more complex cost calculation (see Section 3.3.2.2).

The dual of this problem is equally interesting: find a strategy that minimises the total collection cost with an acceptable level of uncertainty. In that case, the variance $Var[g(Y)]$ needs to be added as a constraint (with some preset value V) to create a meaningful optimisation problem:

Optimisation Problem 3A (Minimise Cost)**.**

$Min_{s \in S}\ \ \sum_{i=1}^{|P|} \sum_{j=1}^{|D_i|} d_{i,j} \times N(D_{i,j}) \times c_{i,j}$

subject to:

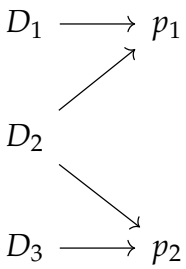
$d_{i,j} \in \{0,1\}$ *for* $i = 1,\ldots,|P|, j = 1,\ldots,|D_i|$

$\sum_{j=1}^{|D_i|} d_{i,j} = 1$ *for* $i = 1,\ldots,|P|$

$Var[g(Y)|s] \leq V$

3.3.2.2 Data Sources That Provide Data for Multiple Parameters

Input-data source mappings have so far been singular and encapsulated from the input parameter perspective. Each input parameter p_i had a list of data sources in $D_{i,j}$ indexed according to the input and their order within that source list. Data sources in our context are an abstract representation of a sampling based collection method, with a number of samples to be collected, and a parameter to populate. It is probable that at least one data source will be able to provide data for more than one parameter at a time. We must define data sources at a higher level than per parameter, but maintain some data quality measures for each parameter (those independent of the data source). There is no preference towards data sources that populate many parameters but these can be more cost effective and should be taken into account.

> **Example.** We continue our slightly simplified supermarket model with one till and two parameters: p_1 mean arrival time and p_2 mean service time. There now exists three data sources without the previous singular mapping.
>
> 
>
>
> D_2 could be a person or software monitoring store traffic that results in estimates for both parameters at the same (with negligible additional cost, significant cost would need to be taken into account and modelled separately). A questionnaire can commonly cover multiple parameters although not well in this case.

Consider now that a data source may map to one or more parameters in P. It is no longer unique to a particular input and must be identified globally. D is a set of all possible data sources. D_j is now a single data source that provides data for one or more input parameters in P. To create strategies one must define which

parameters a source can provide data for. Let d now be a matrix indexed by data source j and input i. d maps valid input to data sources combinations. If data source D_1 can provide data for input p_1 and p_3 then $d_{1,1} = 1$, $d_{1,3} = 1$, and for the others $d_{1,i} = 0$. Viewing over a single source gives an array of boolean values e.g. $d_{1,:} = [1\,0\,1]$.

We still want to optimise the number of samples from a data source. The sample count for data source D_j is denoted as $N(D_j)$ and a strategy s can be simplified to $s = N(D_j), j = 1, ..., |D|$. If $N(D_j) > 0$ the data source is providing samples for one or more parameters as defined by $d_{j,:}$.

For all inputs there is still a data requirement that there must be samples from at least one data source for the strategy s to be valid. This is achieved by the first two constraints in the next Optimisation Problems. No data source's sample size $N(D_j)$ in the strategy can be negative. By using the data source mapping $d_{i,j}$ and sample allocations $N(D_j)$ we check that the total number of samples allocated to each input is greater than zero (across all sources), by doing a cross product of each relevant row in d and the sample allocations array. The sum of each of these products must be greater than zero.

When constraining or optimising by cost, each data source now has a cost function $C_j()$ that does not depend upon a specific input. It takes the number of samples $N(D_j)$ as its only parameter to produce an estimated total collection cost for that data source. The previous Optimisation Problems become:

Optimisation Problem 2B (Advanced Minimise Variance)**.**

$Min_{s \in S}\ \ Var[g(Y)|s]$

subject to:

$N(D_j) \geq 0$ *for* $j = 1, \ldots, |D|$

$\sum(d_{j,i} \times N(D_j)) > 0$ *for* $i = 1, \ldots, |P|$

$T(s) \leq C$ *where* $T(s) = \sum_{j=1}^{|D|} C_j(N(D_j))$

Optimisation Problem 3B (Advanced Minimise Cost)**.**

$Min_{s \in S} \;\; T(s) \quad T(s) = \sum_{j=1}^{|D|} C_j(N(D_j))$

subject to:

$N(D_j) \geq 0 \;\; for \;\; j = 1, \ldots, |D|$

$\sum(d_{j,i} \times N(D_j)) > 0 \;\; for \;\; i = 1, \ldots, |P|$

$Var[g(Y)|s] \leq V$

optional:

$T(s) \leq C$

3.3.2.3 ANOVA

The two previous perspectives have different related issues with regards to the calculated *output variance* $Var[g(Y)|s]$:

- *When the objective is to minimise total cost*, it is difficult to pre-populate the acceptable level of variance V. Especially since this is not the usual type of variance. V must take into account simulation variance if solving by simulation and the accuracy of all the estimators in the process, some of which will be unfamiliar to the user.

- *When the objective is to minimise variance*, if there is no analytic solution available, the optimisation solving algorithm and its accuracy may need to consider and take into account the simulation variance. So far it is assumed that the model simulation and algorithm are run until sufficiently accurate. Each may still have some variability. Once uncertainty from a strategy has diminished, remaining variance is likely to be dominated by any simulation variance. Investing in more expensive strategies will likely see little improvement, possibly even deterioration due to the estimation and simulation uncertainty. We may want to look at minimising $Var[g(Y)|s]$ only to the level that it is indistinguishable from the uncertainty due to simulation and consider minimising total cost as a secondary objective.

Alternative Optimisation Problems using ANOVA (Analysis of Variance) techniques could overcome these challenges for models solved by simulation. Mod-

els solved analytically already differentiate model output variance caused by the strategy, since there is no simulation variance.

Freimer and Schruben [1] explain two overlapping approaches using ANOVA to assess the need for further data samples. One for a fixed effects model and one for a random effects model. Each is presented for single input parameter application then expanded to support multiple input parameters. The process starts with a number of samples for each input parameter, equivalent to a simplified strategy with only one data source per parameter. The model is simulated using experiments that represent the parameter uncertainty given a set number of samples (either using an interval or resampling). ANOVA is then used to decide if the current sample sizes have sufficiently reduced the variance from parameter uncertainty. Sufficient is defined as the variance from parameter uncertainty being indistinguishable from simulation variance (like the earlier motivation). This is decided using an F-test and constrained by a probability of the decision being incorrect. The analysis results are used to decide whether to increment any sample sizes and repeat, or stop process.

Due to the nature of our problem we are interested in the random effects model. The following explains the essentials of the ANOVA part of their approach while integrating our problem formulation. More detailed derivation of certain formulae can be found in most statistics books or in [1].

We first consider the effect of a single input parameter p_1, or factor in ANOVA terminology, on a simulation output. If one performs simulations at a randomly selected factor levels and n simulation replications per factor level (these are equivalent to k and r in Chapter 4), it is assumed the resulting output values can be represented by the following statistical model:

$$Y_{ij} = \mu + \tau_i + \epsilon_{ij} \begin{Bmatrix} i = 1, ..., a \\ j = 1, ..., n \end{Bmatrix} \tag{3.2}$$

Where μ is the overall mean, and τ_i and ϵ_{ij} are random variables. Note here that i (the levels) and j (the replications) are specific to these equations and not related to our own parameter or data source indexing. To be able to test the

hypothesis of the statistical model, we assume that the treatment effects τ_i are independent and normally distributed with mean 0 and variance σ_τ^2, and the errors ϵ_{ij} are independent and normally distributed with mean 0 and variance σ^2. By independence the variance of the output our response is:

$$V(Y_{ij}) = \sigma_\tau^2 + \sigma^2$$

In thesis terms, we can consider σ_τ^2 the variance of the treatment effects τ_i as the variance caused by pushing modelled parameter uncertainty through the model. The null hypothesis for the random effects model, tests σ_τ^2:

$$H_0 : \sigma_\tau^2 = 0$$

This is true if all treatments are identical, and $\sigma_\tau^2 > 0$ if not. The ANOVA decomposition of total variability states [87]:

$$SS_T = SS_{treatments} + SS_E$$

which can be used to formulate a ratio for the test statistic:

$$F_0 = \frac{MS_{treatments}}{MS_E} = \frac{SS_{treatments}/(a-1)}{SS_E/(na-a)}. \tag{3.3}$$

Under the null hypothesis F is a random variable with $a-1$ and $a(n-1)$ degrees of freedom. Using a significance level α, the null hypothesis H_0 is rejected if $F_0 > F_{\alpha,a-1,na-a}$. In the thesis context if the treatments are derived from the parameter uncertainty due to the strategy $X(s)$, and the error variance is due to simulation uncertainty, rejecting H_0 means that $X(s)$ has a distinguishable effect on Var[g(Y)]. Therefore, if $F_0 > F_{\alpha,a-1,na-a}$ we should try to collect more data for the parameter. If $F_0 \leq F_{\alpha,a-1,na-a}$ the effect of further data collection may not be noticeable due to simulation uncertainty.

This is a one-way ANOVA but additional extended options are available to consider the effect of multiple input parameters. Two-way ANOVA for two factors (parameters) up to n-way ANOVA for many factors. When using two-way

ANOVA or greater, one calculates an F-test and p result for each factor (the main effects). One can also calculate results for the interaction effects for groups of factors e.g. p_1 main effect, p_2 main effect, $p_1.p_2$ interaction effect. Freimer and Schruben [1] define the following interpretation of the ANOVA results: If the main effect for a factor (parameter) is significant the factor needs more data collection. If the main effect is not significant, but an interaction effect for two factors is significant, both factors need more data collection. This is further discussed in Section 5.5.

Using ANOVA could breakdown strategy results into contributions of each input parameter. This introduces another per parameter factor into the Optimisation Problem. This new viewpoint fits best with the *minimise total cost* objective but results in a more extreme approach. When using ANOVA, a strategy could be considered invalid if any parameters still have significant main effects (or interaction effects to a sensible degree) when evaluted using $X(s)$. Under the assumption that these parameters need more data.

Optimisation Problem 4 shows the Minimise Cost problem with a new constraint that the strategy does not reject H_0 with some set p value (different from input parameter p). This could be supplementary to the V variance constraint but it would more likely replace it. One may also wish to include an additional constraint, setting a maximum limit on the total cost for valid strategies.

Optimisation Problem 4 (Minimise Cost with F-test)**.**

$Min_{s \in S} \;\; T(s) \quad T(s) = \sum_{j=1}^{|D|} C_j(N(D_j))$

subject to:

$N(D_j) \geq 0 \;\; for \;\; j = 1, \ldots, |D|$

$\sum(d_{i,j} \times N(D_j)) > 0 \;\; for \;\; i = 1, \ldots, |P|$

$\forall p_i \quad F_0 \leq F_{\alpha, a-1, na-1}$ *(first and second order effects)*

optional:

$T(s) \leq C$

The minimise output variance $Var[g(Y)|s]$ perspective is a less suitable case for ANOVA. When solving by simulation this objective is to minimise the variance due to parameter uncertainty. The previous constraint has covered this, within a chosen a probability of being correct. The variance from parameter uncertainty being indistinguishable from the remaining variance means that it has been effectively minimised. If the minimise output variance problem (2B) is kept the same, but with a similar F-test constraint to the previous problem (4), the remaining variability may only be caused by simulation rather than the strategy. If S only contains strategies with indistinguishable variance. We could effectively be ordering by the quality of the simulation result instead of the certainty provided by the strategy.

Going forward the minimise cost approach of Optimisation Problem 4 appears to be the only promising choice for ANOVA usage. It also facilitates iterative improvement based solving, which will be discussed in Chapter 5. The stricter F-test constraint may increase the chance that no valid or optimal strategies are found. In this case one could reduce the significance of the test or re-evaluate other constraints, such as increasing the available budget, after failing to find an optimal result.

3.3.3 Strategy Solution Space

The stategy solution space S (strategy space) is made up of all currently valid data collection strategies in the problem description. A strategy s was previously defined as set of sample sizes $N(D_j)$ for a collection of data sources D. Each data source can provide data for one or more input parameters in P. A strategy includes one chosen data source per parameter. Each unique set of pairings and sample allocations is defined as a different possible strategy.

The complexity of the strategy space is therefore affected by the number of input parameters $|P|$, the number of data sources per input, and the number of sample size levels within a data source. If one considers the complexity of strategy space before any budget or variance constraints, one can estimate the complexity

as:

$$z^{k \times |P|} \tag{3.4}$$

where z is the average number of available data sources per parameter, k is the average number of sample levels per data source, and $|P|$ is the number parameters being populated. The worst case complexity can be considered by replace the averages with the maximum in each case. Both of these are likely to be over estimates in practice as the budget constraint will invalidate many strategies. The valid strategy space will reduce as problem constraints are applied. Depending upon the constraint this reduction from the original strategy space may be before or after model evaluation.

If all data sources in D each have a maximum sample size $N_{max}(D_j)$ or if there is a collection budget C then the entire space can be generated. An exhaustive approach to generating can be conducted by first creating a base strategy for each possible pairing of input parameter to data source, and then generating strategies for valid sample allocations within a base strategy (using the minimum and maximum sample sizes, and sample size increments of the data sources). This can be done recursively and may be additionally constrained using the collection budget or other constraints during generation.

3.4 Summary

This chapter formally defined and explained the problem area, including the type of models this thesis covers, data collection attributes for these models, and optimising the related data collection strategies. This formulation provides a basis for all following ideas and solutions. It defined consistent terms and assumptions that will be used throughout the remainder of the thesis. By breaking down the problem into abstract optimisation problems, we presented an overview of what needs to be solved independently of the solution method. This also allowed us to present the simplest case before expanding to cover important more complex factors from the problem domain.

In the next chapter we will describe ways of representing and using the pa-

rameter uncertainty (and uncertainty of data from data sources) with the model. This covers the model input aspect of providing a solution to these optimisation problems, by addressing the given s in $Var[g(Y)|s]$. Methods for solving the optimisation problems themselves will presented in Chapter 5.

CHAPTER 4

Measuring and Modelling the Effect of Input Uncertainty

In the previous chapter we formally defined the problem domain and created a set of optimisation problems. All of these optimisation problems were based upon the idea of using the model itself to assess the potential improvements provided by different strategies. To enable the model to be used in the evaluation of data collection strategies, we need a way to translate details about the strategy from parameter data quality and data sources into model input parameters as experimental values.

The following chapter explains possible methods of modelling parameter uncertainty as experimental values to be evaluated using the model. These involve three different approaches based on confidence intervals, the Central Limit Theorem, and bootstrap resampling. The parameter uncertainty modelling methods are designed to provide predictions on the behaviour of future data collection based upon past samples and data source attributes. They must also take into account when data for a parameter is coming from multiple different data sources.

Section 4.1 provides an explanation of the motivation for this chapter. Section 4.2 describes a simple confidence interval based method for modelling parameter uncertainty. Section 4.3 builds upon that and uses the Central Limit Theorem to provide our first method. Section 4.4 and Section 4.5 discuss experimental designs that may be used to improve efficiency. Section 4.6 explains how resampling the existing data can be used as an alternative way of modelling the

parameter uncertainty. Section 4.7 discusses the effect of data sources providing data for multiple parameters and how it affects the modelling methods defined in this chapter. Section 4.8 describes how parameter transformation or validation may affect integration. In Section 4.9 we summarise the content and key contributions of the chapter.

4.1 Purpose

This section briefly reiterates more formally the purpose of modelling parameter data uncertainty and the problem we attempt to solve.

Chapter 3 formally defined the modelling and data collection problem. A multi-dimensional random variable $X(s)$ represents the uncertainty of the parameters given a data collection strategy s, where each input parameter p_i is expressed as $X_i(s)$. We defined $Var[E[Y]|X(s)]$ (shortened to $Var[E[Y]|s]$) as the *output variance* as a result of the parameter uncertainty $X(s)$ of strategy s. Since we do not separate it here, this variable may include simulation variance as well as parameter uncertainty variance.

The remainder of this chapter looks at approaches to realising $X(s)$ into model parameters as experiments so that we can attempt to calculate or estimate $Var[E[Y]|s]$. Doing so allows us to compare strategies and begin to solve the optimisation problems described in Section 3.3.2. Where possible we look at limiting the number of values required per parameter since the model solutions may be costly if lengthy simulation is required, and there may be many parameters in the model resulting in a large strategy space.

4.2 Confidence Intervals

A confidence interval is one of the most commonly used ways to describe the certainty of an estimator. To model data uncertainty for parameters we are interested in the confidence interval for a mean. The sample mean of the data samples collected as part of populating an input parameter p_i. This mean will be used

directly, as the value of p_i, or after transformation (see Section 4.8).

For a data source D_1, given an existing sample mean $\bar{X}$ and standard deviation σ, from a number of previous samples m. We can use a 95% confidence interval [5] to calculate low and high values to test for a parameter p_i:

$$\left(\bar{X} - 1.96\frac{\sigma}{\sqrt{m}}\ ,\ \bar{X} + 1.96\frac{\sigma}{\sqrt{m}}\right) \tag{4.1}$$

This gives two experimental values (or treatment levels) for p_i that provide a representation of the certainty of D_1. Additional experiments can be run by repeating the two values equally, r repetitions gives $2r$ experiments. Consider the data source as part of single data source strategy. Solving the model for $E[Y]$ gives $2r$ results for this strategy, and allows us to calculate a mean and more importantly variance $Var[E[Y]|s]$ over the strategy's results.

If we now add a number of new samples n to our sample size (where $n > 0$) as part of a similar strategy, the sample size for D_1 becomes $m + n$. We approximate the new interval as:

$$low = \bar{X} - 1.96\frac{\sigma}{\sqrt{m+n}}\ ,\ high = \bar{X} + 1.96\frac{\sigma}{\sqrt{m+n}} \tag{4.2}$$

Additional samples then cause a reduction in the size of the interval and provide new low and high values for the parameter. We can once again solve the model for $E[Y]$ using these parameter values and compare the $Var[E[Y]|s]$ of the original strategy against this strategy, with additional samples. The process generalises to multiple parameters by pairing all combinations of experiment values e.g. with two parameters { (low, low), (high, low), (low, high), (high, high) }. This generically scales as $2^{|P|} \times r$ experiments, where $|P|$ is the number of parameters being evaluated and r is the number of times each experiment is repeated.

The ability to detect the effect of the intervals will be affected by interval size, the model sensitivity to the parameter, the intrinsic uncertainty of the model, and the number of model solutions used. Solving the model analytically would allow the effect of the parameter variation to be much more noticeable, since the simulation uncertainty is removed. With simulation-based solving the resulting variance is made up of simulation variance and parameter uncertainty variance. We

can either look at separating the two variances, or more likely, use indistinguishable parameter uncertainty variance as a goal of the strategy variance reduction. Ideally, we increase the samples allocated to the data sources until the intervals no longer produce notable improvement. Freimer and Schruben [1] complete this approach in more detail, including defining the required number of experiment repetitions. Therefore we do not continue it further but it provides a good motivation for the idea of modelling parameter uncertainty through the model.

The technique described provides only two extreme values to represent the parameter data uncertainty. We must asssume the output response in monotone over the interval, or by discounting values in between it may ignore some behaviour of the model output. What if we wanted to remove this assumption using many values, something that covered more of the distribution? A more complete representation of the space would also allow experimental design approaches beyond all combinations of low and high. In the next section we provide a method that offers this.

4.3 Normal distribution-based

This section describes an approach to modelling parameter data uncertainty based on the Central Limit Theorem and Normal distribution sampling. We also discuss the usage of design of experiments to cover the problem space more efficiently.

Our approach would allow any characterisation of $X(s)$ to indicate uncertainty in the input parameter values. However, if sources consist of samples that are averaged to estimate an input parameter value, it is very natural to exploit sampling statistics to gain insight in the variability of the source. In particular, let $x_1, x_2, \ldots, x_N$ be samples, then the *sample mean* and *sample variance* are given by [87] :

$$\widehat{E[X]} = \frac{1}{N} \sum_{n=1}^{N} x_n, \tag{4.3}$$

and

$$\widehat{Var(X)} = \frac{1}{N-1}\sum_{n=1}^{N}(\widehat{E[X]} - x_n)^2. \tag{4.4}$$

The Central Limit Theorem [87] then says that the distribution of the sample mean $\widehat{E[X]}$ as a function of N converges to:

$$\widehat{E[X]} \rightarrow \mathcal{N}(\widehat{E[X]}, \sqrt{\frac{\widehat{Var[X]}}{N}}), \tag{4.5}$$

where $\mathcal{N}(\mu, \sigma)$ is the Normal distribution with mean μ and standard deviation σ. It is important to stress that the use of the Normal distribution around the sample mean does not imply that we assume the model incorporates Normal distributions. For instance, if we have a model with an exponentially distributed delay for some event, we would use the sample mean to estimate the rate of the exponential distribution. There is no Normal distribution in the model, and the Central Limit Theorem applies and represents uncertainty in the estimate for the rate of the exponential distribution.

We use the Central Limit Theorem as follows, focusing initially on Optimisation Problem 1. In that formulation of the optimisation problem, a strategy is given by $s = \{N(D_{i,j}\}, i = 1, \ldots, |P|, j = 1, \ldots, |D_i|$. It may be that some data has been collected already, namely $M(D_{i,j})$ samples, and that the sample mean is $\hat{\mu}_{i,j}$ and sample variance is $\hat{\sigma}^2_{i,j}$. Then for source $D_{i,j}$ we expect that if we add $N(D_{i,j})$ samples we obtain the following Normal approximation of the sample mean (applying the Central Limit Theorem as indicated above):

$$\mathcal{N}(\hat{\mu}_{i,j}, \frac{\hat{\sigma}_{i,j}}{\sqrt{M(D_{i,j}) + N(D_{i,j})}}). \tag{4.6}$$

If possible, $\hat{\mu}_{i,j}$ and $\hat{\sigma}_{i,j}$ are estimated from the $M(D_{i,j})$ initial samples using the sample mean and variance as in Equation 4.3 and Equation 4.4. Of course, if $M(D_{i,j}) = 0$ there are no existing samples, in which case one needs to use best effort to determine $\hat{\mu}_{i,j}$ and $\hat{\sigma}_{i,j}$. The essence of our approach is that we model the uncertainty associated with the data sources (see Section 5.7).

The main observation now is the following equality, in which $f_{X(s)}(x)$ is the

probability density function for the random variable $X(s)$ (as in Equation 3.1), and $g(Y(x))$ denotes the output of the model for input x :

$$E[g(Y)|X(s)] = \int_x g(Y(x)) f_{X(s)}(x) dx. \tag{4.7}$$

The equation says that the output $g(Y(x))$ needs to be computed over all possible input parameter values of the strategy $s \in S$, as drawn from the distribution of $X(s)$. We are now in a position to connect input parameter uncertainty with output uncertainty by solving the model based on samples drawn from the Normal distributions that express the parameter uncertainty. We will have to do that for every feasible strategy and then select the optimal strategy based on Section 3.3.2. The first solving algorithm using this technique is presented in Section 5.2.

4.4 Stratified Sampling

If we consider the uncertainty modelling proposed in the previous section, fairly sampling all possible values of the distribution of $X(s)$ is impractical for all but the simplest models. It would produce too many model experiments to solve. Instead, we randomly sample the distributions to provide an approximation for the expect value $E[g(Y)|s]$. The quality of the approximation depends upon the number of times the distribution is sampled and any experiment design approach used. Note that this sampling and quantity is distinctly different from samples with regard to data collection. The distribution sampling contributes towards the accuracy of strategy results, rather than data collection certainty, and varies independently.

With a sufficient number of experiments, and therefore distribution samples, an accurate approximation could be represented. Where possible, we wish to limit the number of model solutions one needs to compute, as this is expected to be the most costly factor in the optimisation solving. Rather than completely independent random sampling of the distribution of $X(s)$, one can use design of experiments techniques to attempt to better cover the space, even with a small

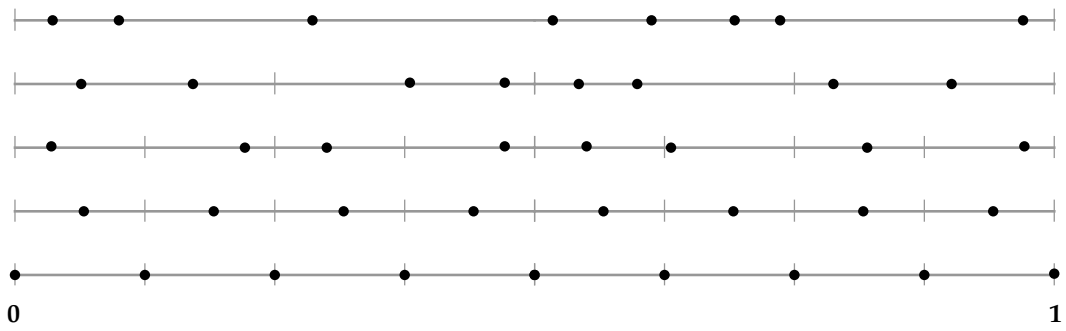


Figure 4.1: Different ways of distributing points using intervals [14]

number of samples.

In general, stratified sampling consists of splitting portions of samples amongst groups within a population so that groups are proportionally represented. In the context of testing input values, stratified sampling is an experimental design approach that can improve the rate at which estimated results converge to their true values [14]. By splitting a distribution into equally probable intervals that contain an equal number of samples, the analysis can better cover the input space with smaller sample sizes compared to completely random sampling.

Samples are randomly distributed within an interval or positioned according to a design. Figure 4.1 show examples of possible interval based sampling for a single parameter. Each interval is of equal probability and each dot is one sample. The *top line* is not split and randomly sampled, equivalent to one interval. The *second and third lines* show random sampling within intervals, each interval contains one and two samples respectively. In the *fourth line* each interval contains one sample, which is positioned in the midpoint of the interval. This results in a biased estimate for variance. The *bottom line* is split into the same intervals but the samples are positioned at the interval end points resulting in one extra sample. This includes the complete range of parameter values.

Endpoints-based design needs to be adapted if the uncertainty modelling distribution has very long tails, since the value of the first and last point may be emphasised while outside the expected or valid parameter range. For this reason we use only random sampling within intervals and midpoints going forward, other options exist for future work. The implementation in Appendix A primar-

ily concentrates on random sampling within intervals, but it also offers interval midpoint sampling for more predictable sampling behaviour and testing.

The stratification can be applied to the Normal distribution based approach to representing the parameter uncertainty. By splitting the normal distribution into intervals with equal probability we can guarantee coverage of distribution even with a small number of experimental values. When mapped to the probability density function of a Normal distribution the intervals would no longer be of equal length. Highly probable areas around the mean in the centre will have smaller intervals and the intervals will rapidly increase in size as you go towards the tails.

Let D_1 be data source with existing data $\hat{\mu} = 5, \hat{\sigma}^2 = 10$, $M(D_1) = 50$, and $N(D_1) = 0$. Generating four values using plain CLT normal distribution sampling and then using 4 intervals both random and mid points results in:

Table 4.1: Values For a Parameter Using Random and Stratified Sampling

Random Unsorted	Random Sorted	Interval Random	Interval Midpoints
19.0414	19.0414	19.2429	19.2981
19.8435	19.3666	19.5582	19.7128
19.5446	19.4637	20.1113	20.0000
19.4637	19.5446	20.4397	20.2872
19.3666	19.8435	20.6963	20.7019

4.5 Experimental Designs

To solve the model, each parameter in P requires a value. We refer to each complete set of one single value per parameter as an experiment. Parameter values modelling uncertainty are generated independently (per parameter) and one must choose how to combine these into experiments. When the Normal distributions representing $X(s)$ are sampled randomly one can simply use the order the values are generated to create one experiment per value generated. Where the number of experiments R is equal to k the different values used to represent

parameter uncertainty, the number of times one samples from each parameter's distribution.

Confidence Interval based uncertainty modelling in Section 4.2 briefly introduced pairing all combinations of input parameter values. For example, two parameters with two values each $(low, high)$ became four experiments $\{(low, low)$, $(low, high)$, $(high, low)$, $(high, high)\}$. This is commonly known as 2^k factorial design (where k here is the number of parameters). Generically, using all combinations, the number of experiments can be calculated in our terminology by:

$$R = k^{|P|}. \tag{4.8}$$

Stratified sampling using intervals generates a sorted or partially sorted set of values for each parameter. Translating these into experiments requires a design. The order can be randomised but this does not guarantee complete coverage of the uncertainty space. The stratification would only guarantee coverage on a per parameter level. Using all combinations of parameter values provides the ideal solution to covering the parameter uncertainty space but at a greatly increased cost in terms of the number of experiments.

Using all combinations is formally called a full factorial design, where there is an experiment for every combination of input (factor) values. Let |P| be 2. If we limit the solving to $R = 25$ experiments, when using basic random sampling the number of values per parameter $k = 25$. If we use all combinations, we are only able to sample $k = 5$ values per parameter. All combinations clearly becomes costly as the number of parameters increases.

These options represent the two extremes of experiment design. Alternatives exist, such as latin hypercube sampling [34], that balance the total number of experiments used against the number of values used per input. The alternatives try to efficiently cover a lot of the input space fairly but not necessarily all of the extremes.

4.6 Bootstrap Resampling

This section discusses an alternative approach to parameter uncertainty modelling using the existing data and resampling, called bootstrap resampling. In its standard usage, bootstrap resampling attempts to discover more information about the behaviour of a test statistic without collecting additional samples [88]. By resampling with replacement from an existing set of real world samples we can learn about the distribution of an estimator e.g. the mean.

This bootstrapped estimator is calculated by the following process, using a set of samples $\{x_1...x_m\}$:

1. Create b bootstrapped sample sets of size m by resampling with replacement m times.

2. Compute the test statistic e.g. mean over each bootstrap resulting in b bootstrapped estimators.

The distribution of these new estimators can be analysed using common techniques e.g. bootstrapped confidence interval. In stochastic modelling this has been used to look at how uncertainty in the input model, from fitting to samples, can affect uncertainty in outputs and performance measures.

Bootstrap resampling can be used as a method for represent parameter data uncertainty [1]. Bootstrap resampling requires existing samples, so all data sources under consideration must have existings samples, rather than summary statistics (mean, variance, sample size). Let the samples used in resampling be the data source's existing samples for p_i. Sample size $m = M(D_j)$. To model different data collection sample sizes, we resample $M(D_j) + N(D_j)$ times for each bootstrap, instead of the original m. We bootstrap a mean estimator, which can be used as single value for p_i. The number of bootstrap estimators required becomes k, the number of experiment values generated per parameter to represent $X(s)$.

First consider the simplified problem, where data sources are specific to single input parameter. One can use bootstrap resampling to create experiment values for the parameters using the data source of each parameter.

```
do{
  for each D_{i,j} with M(D_{i,j}) + N(D_{i,j}) > 0 {
    resample M(D_{i,j}) + N(D_{i,j}) times to get B*_{i,j}
    x_{i,j} = E[B*_{i,j}]
  }
  set x = ({x_{i,j}});
  use x as required...
} k times (or a variable amount depending upon the solving algorithm)
```

For each experiment, a sample set is resampled, once for each parameter. The mean is taken over the bootstrapped samples and used for the associated parameter with the model. The version where data sources can map to multiple parameters is slighlty more complicated since one may need to combine data as we will discuss in the next section (Section 4.7). The following algorithm presents one option. For each data source providing data for a parameter p_i, one creates a resampled data set and a bootstrapped estimate. These are then combined to produce an experimental value for the parameter p_i.

```
do{
  for each p_i
    for each D_j where d_{j,i} = 1 and M(D_j) + N(D_j) > 0 {
      resample M(D_j) + N(D_j) times to get B*_{i,j}
      x_{i,j} = E[B*_{i,j}]
    }
    set x_i = E[x_{i,j}]
  }
  set x = ({x_i});
  use x as required...
} k times (or a variable amount depending upon the solving algorithm)
```

4.7 Combining Active Data Collection

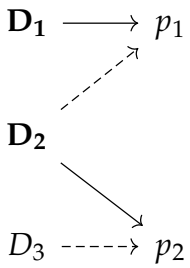
If an input parameter has multiple sources of data, these need to be taken into account and possibly combined prior to populating the parameter. This section explains why this may occur and different approaches to resolution. Section 4.7.1 provides motivation for the section with a problem definition and example causes of multiple data sources. Section 4.7.2 discusses existing statistical approaches and assumptions made regarding their usage. Section 4.7.3 explains the solution as different data combining and selection modes including how these relate to the previous uncertainty modelling methods in Section 4.3 and Section 4.6. Section 4.7.4 summarises the section including what will be used going forward.

4.7.1 Definition and Causes

The model requiring data has a set of input parameters P to populate. There is a set of all available data sources D. In the expanded optimisation problem, each data source D_j can provide data for one or more input parameters simultaneously. For example a complex measuring process could provide data for only one parameter, whereas software monitoring solution may measure multiple system attributes resulting in data for multiple parameters. If the cost of providing data for two parameters instead of one is negligible the collection is modelled as a single source. For each input parameter p_i there exists an array of available data sources, and as part of a collection strategy s, p_i has one or multiple *active data sources*.

These capabilities result in the relationship between input parameter and data source being one-to-one in the simplest case (a source capable of providing data for only one parameter, and being used only by that parameter) up to a many-to-many (an input parameter receiving data from multiple data sources, and one of those data sources providing data to multiple parameters). Ideally in a collection strategy each parameter p_i has one chosen active source, the one that provides the best possible data. Even then there can be multiple sets of data available to use for the parameter.

Example. Here we expand upon our example from Section 3.3.2.2, a simplified supermarket model with one till and two parameters: p_1 mean arrival time and p_2 mean service time. There are three data sources and a strategy selects D_1 to provide samples for p_1 and D_2 for p_2. D_2 simultaneously (as a bonus) provides samples for p_1



Given the problem formulation, we now have two sets of data source information for p_1 containing at least: $\{\hat{\mu}_{1,1},\ \sigma\mu^2_{1,1},\ M(D_1),\ N(D_1)\}$, $\{\hat{\mu}_{2,1},\ \sigma\mu^2_{2,1},\ M(D_2),\ N(D_2)\}$. We must decide whether and how to use the secondary data for p_1. Modelling the uncertainty of future data collection has so far been orientated for a single data set.

In an ideal scenario, without constraints, D_1 can be provide as much of the 'best information' as required so the additional data samples are redundant and can be ignored without any effect. As soon as any constraints are added on the available collection resources, such as a maximum sample size on D_1, it becomes important to properly evaluate the usage of data provided by multiple sources simultaneously.

4.7.2 Statistical Approaches

Many statistical techniques for comparing two (or more) data sets already exist. Common statistics such as the sample mean, median, standard deviation, and sample variance [89] are taken over a single set of samples but can be used to compare different data sets. To decide how to use two sets of samples in this case we are more interested in confirming both have the same true mean and where appropriate a similar distribution. Statistics usually offers the inverse, the ability

to predict if they are significantly different with a chosen level of confidence. These techniques can analyse for example, whether two sets of samples are likely to have the same true mean or if the samples have statistically similar variance, such as Student's t-tests, F-test [89], or Levene's test [90].

There are two useful results of using the statistics mentioned previously: one can further analyse and validate the data provided by a single data source and one can estimate if the data produced by two different sources (for the same parameter) is significantly different. The second is important for this section. If we are populating an input parameter we should obviously not use and combine significantly different data sets. Separate of this we must also question why two different data sources for the same parameter can or have produced significantly different results.

This problem goes into validating a data source (collection method) as suitable for populating an input parameter, which is outside the scope of the thesis. For our usage we assume statistical techniques have been used to evaluate the existing samples collected for each data source in D. We assume the result is that, for each input parameter, the data sources do not have significantly different true means. In practice it means that information from the sample sets could be combined but this is not trivial. In the next section we discuss how they could be combined in a way suitable for the methods of Section 4.3 and Section 4.6.

4.7.3 Combining & Selection Modes

In the previous sections it was explained how and why multiple data sets per input parameter can occur and statistical approaches to comparison between sets. Even if these data sets are not considered significantly different it is often not appropriate here to just combine them into one larger set of samples. This is due to the fact we are not just looking at the existing samples, we are also making some predictions about the future samples from each respective data source. The estimations, as part of modelling parameter uncertainty in Section 4.3 and Section 4.6, are based on per source attributes including the total number of samples to be collected. These therefore apply to and affect the data sample sets indepen-

dently rather than as a combined set.

If conducted, the combination phase must be completed after modelling the future data collection. There are different ways this can be achieved. The rest of this section discusses data combining and selection modes: different options for combining the information or deciding which to use in our parameter uncertainty modelling. This includes how they can be used with methods from Section 4.3 and Section 4.6, their default modes, and potential conflicts with optimisation solving methods discussed later in Chapter 5.

4.7.3.1 All Data

The most obvious mode is *All*, attempting to model all active data collection within the parameter uncertainty modelling under the assumption that all data will be used after collection. This requires combining quality measures of the active sources (for a parameter) before, and modelling their uncertainty once, or by modelling the data uncertainty once for each source and feeding the combined results into the system model. Either way the combination will likely involve weightings based upon the amount of data contributed by the source. For each input and data source we define a weight $w_{i,j}$.

$$w_{i,j} = \frac{M(D_j) + N(D_j)}{A_i}$$

The weight for an active data source is the total number samples for the data source, existing $M(D_j)$ and new $N(D_j)$) over A_i the total number of samples for the input from all active data sources. This weighting is one way of taking into account the number of future samples when combining current data source information.

If we first consider the Central Limit Theorem-based Normal distribution sampling proposed in Section 4.3, it takes estimates for the sample mean, standard deviation, and total number samples for a source. From this it can produce values for the input parameter, by sampling from the obtained Normal distribution, these represent the uncertainty of the data for model-based evaluation. While

sharing a (target) population mean, these data quality measures and attributes are source specific. Completing this uncertainty modelling per data source for an input parameter would result in separate experiment sets and separate results. We could weight these model results to better represent the combined data but this would be inefficient (for a parameter with two active sources). Since the model execution complexity is already expected to be an important consideration [14], we need to combine the data source information going through the uncertainty modelling prior to model execution.

If we assume after collection that the two (or more) sets of data samples are combined, we can assume the distribution of the combined collected data samples will be similar to a weighted combination of our estimates. More specifically, we weight the data source's estimated sample mean and standard deviation while combining them for use in the normal distribution. This results in the following equations when using more than one active data source and *All* combination mode.

$$\mu_i = \sum_j \left(w_{i,j} \times \frac{\hat{x}_{j,i}}{\sqrt{M(D_j) + N(D_j)}} \right)$$

$$\sigma_i = \sum_j \left(w_{i,j} \times \frac{\sigma_{j,i}}{\sqrt{M(D_j) + N(D_j)}} \right)$$

$$\mathcal{N}(\mu_i, \sigma_i) \tag{4.9}$$

Bootstrap Resampling described in Section 4.6 involves resampling the existing samples from a data source a number of times. The mean is then taken over the resamples to create one value for the input parameter. In our case the number of resamples is equal to the total number of samples allocated to a data source as part of a strategy. Using *All* data with two (or more) active data sources, a single bootstrap resampling results in two sets of bootstrapped samples B^*. Since the number of samples in each of the bootstrapped samples already matches the total number of samples for each data source $M(D_j) + N(D_j)$, we can combine the bootstrapped samples without weighting $B_i^* = \{B_{i,j}^*\}$. This produces one

representation of the complete data from two sources after collection. The mean is taken over the combined set of samples, rather than each set of bootstrapped samples, and provides a single experiment value for the input parameter p_i. For multiple experiments the entire process is repeated resulting in a new sets of bootstrapped samples and another mean estimate. The *All* data combining mode is the most suitable for use with Bootstrap Resampling and is therefore the default in the implementation (Appendix A).

4.7.3.2 Best & Worst Data

Most of the remaining modes described here result in a decision about which data to use from the multiple active sources, rather than strictly combining. These cover the key perspectives: *best*, *worst*, and *independent of the data*.

If we have more than one active data source for an input parameter, it is logical to consider choosing only the data source which provides the 'best data'. In the problem definition presented here there are two ways of defining the best data: the data source with the best existing data samples (*Best Existing*) and the data source predicted to provide the best data from future collection (*Best Estimated*). The existing samples approach looks only at the samples already collected. This would not take into account the amount of samples expected to be collected as part of the strategy, which makes it less beneficial in practice. If you have two data sources providing data for an input parameter (D_{pricey}, D_{cheap}), D_{pricey} has better existing data than D_{cheap} but costs a lot more to collect. There may exist scenarios where little or no further data could be collected with D_{pricey} but lots of additional samples could be provided by D_{cheap}. When combining or selecting using *Best Existing*, D_{pricey} would still always be used in the modelling, resulting in no obvious improvement in parameter certainty even while varying the samples allocated to D_{cheap}.

Best Estimated tries to use the parameter uncertainty modelling method (or other analysis) to decide which data to choose when multiple active sources are available. For example, one could estimate using the Central Limit Theorem (CLT) similar to Section 4.3 by estimating the variance of the future samples

(given $N(D_j) + M(D_j)$). One would then choose the data from the data source predicted to produce the least variance and the most confident mean estimate to model in the strategy. This approach is highly dependent on the parameter uncertainty modelling method but it takes into account the future sample sizes rather than just existing data.

Best Estimated mode works best with CLT normal distribution-based parameter uncertainty modelling since it gives a single comparable value. While it can work with bootstrap resampling, it is more complex to calculate. One possibility is to look at the distribution of the mean from the resampling by repeating the resampling process many times for each active data source. The quality of this estimate is then also based upon the number of repetitions used here, adding further uncertainty compared to straight calculation.

An inverse approach is to model the parameter uncertainty expecting the *worst*. If for a parameter one used the total sample count over all active data sources A_i and the worst existing data's sample mean and variance in the parameter uncertainty modelling, one could then assume that the collection strategy is likely to provide this $Var[g(Y)|s]$ or better (less variance). Since some of the data in the strategy is coming from data sources known to be better than their approximate combined representations. Apply the above theory, the potential benefits of better data sources become hidden. This makes it difficult for the optimisation algorithm to find the real optimal solution(s). As an extreme example: consider a strategy s where the worst data source provides data for all the input parameters. In this mode regardless of the other selected data sources and sample sizes, the sample data from the worst data source would be used in parameter uncertainty modelling. It would then dominate and poison the strategy evaluation and attempts to optimise. Given these drawbacks this mode is not recommended and left only as a thought exercise.

4.7.3.3 User Decides

Human intervention is a simple option to solving this data combination or selection problem, either as a primary or fallback option. The user could be asked to

pre-specify data combining logic or select a primary source when needed (from the parameter's active data sources). This externalises the problem to a *user decision*. Custom data combining logic would require advanced understanding and implementation changes so may be beyond many optimisation users. The more usable option is allowing the user to define the primary data source for parameters with multiple active data sources. This effectively overrides any other data combining or selection mode. Data from the primary data source is then the only data and source information used in the parameter uncertainty modelling (for that parameter).

This mode works best when the user is certain about the data source they wish to use (or not use) for a parameter. In a way it is a user constraint on the optimisation problem, restricting the data options under certain conditions. It could be done on demand or singular using user specified strategies. That would become cumbersome with many overlapping data sources or with many strategies under evaluation. Rather than specifying per strategy, the user could be asked to specify a data source preference ordering either over all sources or on a per input basis. This would only be used for solving the problem in this section and would not affect overall data collection optimising.

4.7.3.4 Test All Independently

The final option explored here is the exhaustive approach: *Test All*. With two active data sources in the strategy providing data for one parameter, one could use the source information separately and model the parameter uncertainty once for each. The result is two separate sets of experiment values for the input parameter, one for each active data source configuration. These could be separately combined with the experiment values from the rest of the strategy and evaluated with the model independently. One would effectively be separating it into multiple strategies or sub-strategies, evaluating each single source combination within the current strategy and then receive strategy analysis results for each. This could require twice the model computation and rapidly multiplies with additional active source decisions, as one would need to test all combinations of

internal strategy choices.

The results would be used to try to decide which data should used for the parameter or in further analysis to try and estimate the effect of using all data in the strategy. If it is just used to decide, this option becomes very similar to the simplified problem, without the ability for parameter data to come from more than one source simultaneously. Re-combining the strategy results to represent using all data would be difficult, especially with more than one input parameter with multiple active data sources.

4.7.4 Summary

The data combining and selection modes described in this section try to explore plausible approaches for solving the problem of integrating information from multiple active data sources into the proposed parameter uncertainty representations. Multiple active data sources are the result of two or more collection methods in the strategy providing data samples for the same parameter simultaneously. Each mode has their own benefits and drawbacks and can work differently with the earlier methods in this chapter so there is no single best or guaranteed to be the most effective mode. Multiple will be made available in the implementation to provide the choice and method appropriate defaults.

The following modes will be included *All*, *Best Estimated*, and *User Decides*. *All* mode is the ideal solution where all collected samples could provide a benefit. It also integrates well with Bootstrap Resampling. *Best Estimated* provides a good alternative and can provide a single selection. *User Decides* allows manual override and externalises the problem should the user disagree with other techniques. The others are not further explored. *Worst* mode makes solving the optimisation problems more difficult as it hides the effect of additional data collection. *Test All* becomes very similar to ignoring the subproblem altogether and increases complexity. The modes will not be heavily evaluated against each other in the results chapter, as the mode chosen is not expected to have a significant effect on the examples.

4.8 Data Transformation & Validation

Certain types of input parameters will not always map directly to the data produced. The data needs to be *transformed* to be usable with the parameter in question. For example, an arrival rate parameter λ could have a data source that collects data to estimate the mean time between arrivals. The transformation in this case is trivial but its inclusion and positioning in the process is still important. The data source uncertainty modelling represents the original untransformed state. It models the mean arrival time. The transformation to a rate must be conducted on each produced value prior to usage as the model's parameter.

4.8.1 Parameter Value Checking

After any transformations it is essential to check whether the generated parameter value is within the valid range for the parameter. While many of the uncertainty representations proposed in this chapter will result in testing small changes in input parameters, it remains possible that the values generated can come close to or be outside the valid range. To keep within the validated range of the model, each input parameter can have a lower and upper limit, and other requirements that need to be adhered to or else the model output may be invalid. Some limits are common to specific data types such as probabilities being restricted to between 0 and 1, a rate parameter being greater than zero, or a parameter requiring only integer values.

If not taken into account by the uncertainty modelling technique invalid values need to be filtered and re-generated to fairly represent the remainder of the valid space. The filter and re-generation must be monitored since too many re-generations shows either: not enough data (since there is too much uncertainty), an issue with parameter uncertainty modelling, or an issue with the models design for that input.

4.8.2 Experiment Checking

Some models include dependencies between parameters, such as $p_1 < p_2$. These can be checked on a second pass of filtering. The experiment values can be regenerated if randomly sampled such as using CLT normal distribution sampling or random interval-based sampling. When using more strict experiment designs, for example using all combinations parameter values, re-generation is not as simple and is likely ineffective. The experiment or experiments in the invalid parameter space need to be removed prior to strategy evaluation. A strategy result made up of less experiments needs additional interpretation, and would often make the strategy sub-optimal regardless of the result.

Where practical these dependencies could be included and avoided within the uncertainty modelling, but that would introduce additional complexity. Another alternative would be to alter the model abstraction to remove the dependency or reduce the likelihood of its occurrence. The current solving algorithms will rely only upon regeneration and deletion, we leave the other options to future work.

4.9 Summary

In this chapter we discussed the problem of modelling parameter data uncertainty in a way which can be used with the model. Feeding this representation through the model and into model outputs allows us to take into account the model's sensitivity to input parameters and the parameter uncertainty from the strategy as a whole. In the final sections of the chapter we discuss other issues that may arise during parameter uncertainty modelling, considering the effect of valid parameter ranges and how using data from multiple different sources at once affects the parameter uncertainty modelling.

In the next chapter we describe how these parameter uncertainty modelling methods can be used to evaluate data collection strategies as part of the larger optimisation solving algorithms.

CHAPTER 5

Solving by Simulation

This chapter describes six algorithms for solving the optimisation problem of efficient data collection for model parameters. These algorithms take valid strategies and provide recommendations on the optimal strategy or strategies depending upon the optimisation problem objective and constraints. All the solving algorithms use the model to evaluate the strategies by testing the effect of parameter uncertainty on one or more outputs.

Section 5.2 presents the a generic algorithm overview and the Basic Exhaustive Algorithm. Section 5.3 details a solving algorithm combing the Basic Exhaustive Algorithm and importance sampling. Section 5.5 explains the Iterative Algorithm, which uses ANOVA and selective strategy improvement to find a solution. Section 5.6 describes issues that can occur due to a large problem space and how these can be overcome by intelligently exploring a subset of the problem space. Section 5.7 discusses any assumptions and limitations of the algorithms detailed in this chapter. Section 5.8 summarises the content and key contributions of the chapter.

5.1 Introduction

In Chapter 3 we defined a variable $Var[g(Y)|X(s)]$ (shortened to $Var[g(Y)|s]$), the variance in the output measure $g(Y)$ due to the parameter data uncertainty in the data collection strategy s. Chapter 4 described different methods of repre-

senting the data uncertainty $X(s)$ within model experiments for execution and evaluation. This provides the input part of solving the optimisation problems. The following sections provide methods for turning experimental values representing $X(s)$ into an estimate for $Var[g(Y)|s]$ of a strategy. The solving algorithms attempt to find the optimal strategy (or strategies) in the solution space but may not evaluate $Var[g(Y)|s]$ for all strategies. The solution space and objective of the optimal strategy are defined in Section 3.3.2.

To recap important terminology: We are optimising the data collection over a set of model parameters P using a set of data sources D. Each data source D_j can provide data for one or more p_i parameters. A data source is made up of multiple properties (see Section 3.2.2, for solving one is especially interested in the number of existing samples $M(D_j)$, the number of new samples $N(D_j)$, and the collection cost $C_j(N(D_j))$. The existing samples provide an estimate for one or more parameters $\hat{\mu}_{j,i}$ and variance $\hat{\sigma}^2_{j,i}$. A strategy $s = \{D, N(D_j), T(s)\}$ and provides a configuration for data collection, specifying a unique combinations of $N(D_j)$ new sample sizes at a total strategy cost $T(s)$.

Similar to many sensitivity and uncertainty analysis methods[14, 91], the solving trys to be 'model-free' by taking a black box approach to the model. It attempts to use only information about the inputs and observed outputs to be as generically applicable as possible.

5.2 Basic Exhaustive Algorithm

This section explains our first optimisation solving algorithm known as the Basic Exhaustive Algorithm. The algorithm uses uncertainty modelling methods from Chapter 4 to analyse every valid strategy before comparing the results.

The simplest, naïve approach to solving can be achieved by generating all the valid strategies in S and then evaluating $Var[g(Y)|s]$ for every strategy. One creates an estimate of $Var[g(Y)|s]$ using model solutions for experiments that represent $X(s)$. The experiments are generated with a parameter uncertainty modelling method from Chapter 4, here we use CLT normal distribution sampling

and bootstrap resampling but others could be used.

Optimisation Algorithm 1 specifies the algorithm independent of the parameter uncertainty modelling. Prior to the for loop (line 2) one defines all currently valid strategies S based on the constraints, such as the collection budget, input-data source combinations, and sample size restrictions. It is the currently valid S since constraints may depend upon the result of $Var[g(Y)|s]$, strategies may later become invalid solutions. One then creates the $Var[g(Y)|s]$ estimate by repeatedly generating parameter values (lines 4 to 7), solving the model (line 8), and iteratively updating the estimators. The process is repeated until the algorithm results are sufficiently accurate, which may be a set number of runs or based on the confidence estimate (see Section 5.4). One must recheck which strategies in S still meet the constraints, before evaluating the objective function of the optimisation problem, either minimise $Var[g(Y)|s]$ (output variance) or minimise $T(s)$ (total strategy cost).

Optimisation Algorithm 1 (Basic Exhaustive Algorithm)**.**

1. define S based on constraints (without needing results);
2. for each $s \in S$ {
3. do {
4. for each p_i {
5. generate x_i using active data sources for i in D
6. }
7. set $x = (\{x_i\})$;
8. solve $y = g(Y(x))$;
9. update $E[g(Y)|s]$ using y;
10. (Eq.(4.3) with y for x_n)
11. update $Var[E[g(Y)|s]]$ using y;
12. (Eq.(4.4) with y for x_n)
13. }
14. until $Var[E[g(Y)|s]]$ accurate
15. }
16. filter S based on any constraints requiring $Var[E[g(Y)|s]]$;

17. select s that minimises $Var[E[g(Y)|s]]$ or minimise $T(s) = \sum_{j=1}^{|D|} C_j(N(D_j))$;

The algorithm can be customised to the Optimisation Problem and parameter uncertainty method used. Optimisation Algorithm 2 presents an algorithm to solve Optimisation Problem 1 (Sample Constraint) using CLT normal distribution sampling. Similar algorithms can be formulated for Optimisation Problems 2A to 3B.

Optimisation Algorithm 2 (Basic Exhaustive Algorithm for Problem 1)**.**

1. define S based on constraints (without needing results);
2. for each $s \in S$ {
3. do {
4. for each $D_{i,j}$ with $M(D_{i,j}) + N(D_{i,j}) > 0$ {
5. draw $x_{i,j}$ from $\mathcal{N}(\mu_{i,j}, \frac{\sigma_{i,j}}{\sqrt{M(D_{i,j})+N(D_{i,j})}})$;
6. }
7. set $x = (\{x_{i,j}\})$;
8. solve $y = g(Y(x))$;
9. update $E[g(Y)|s]$ using y;
10. (Eq.(4.3) with y for x_n)
11. update $Var[E[g(Y)|s]]$ using y;
12. (Eq.(4.4) with y for x_n)
13. }
14. until $Var[E[g(Y)|s]]$ accurate (Section 5.4)
15. }
16. filter S based on any constraints requiring $Var[E[g(Y)|s]]$;
17. select s that minimises $Var[E[g(Y)|s]]$;

Note that the algorithm purposely avoids the border case $M(D_{i,j}) = N(D_{i,j}) = 0$ for a source $D_{i,j}$ (line 4). This case corresponds to the situation that for source $D_{i,j}$ we have no earlier samples ($M_{i,j} = 0$) and collect no additional samples

($N_{i,j} = 0$), but we do have an assumption about the sampling mean and standard deviation ($\mu_{i,j}$ and $\sigma_{i,j}$). In our proposed approach, we must rank these strategies as leading to infinite variance in the input. Thus we are not able to compute variance of the output for this strategy, and draw the justified conclusion that such a strategy is inferior to any other strategy. Finally, we note that the set S of possible strategies is derived from the distribution of the N available samples over all sources. This can be programmed conveniently with a recursive algorithm, the details of which are not included here.

All subsequent algorithms are specialised, more advanced versions of this algorithm, which use alternative approaches to either: calculate $Var[g(Y)|s]$ or search through the strategies S (instead of exhaustively calculating and comparing all $Var[g(Y)|s]$).

5.3 Importance Sampling Extension

This section presents a modified version of the Basic Exhaustive Algorithm. By using importance sampling, the modified version can require less model solutions per algorithm execution than the original.

The Basic Exhaustive Algorithm (Algorithm 1 & 2) may be expected to be time consuming, since for each strategy the conditional output variance $Var[E[g(Y)|s]]$ needs to be computed. Naïvely, one could do this by simply running the models many times, for enough samples of the Normal distributions that represent the input parameter uncertainty, but in this section we will show that samples can be weighted to obtain results for many strategies concurrently. The idea of weighting is identical to importance sampling, so we named our approach after that technique.

Let us assume a target strategy $s \in S$ will be analysed by reusing the results of *anchor* strategy $s_a \in S$. Translating the importance sampling weight into our strategy and model terms, let the weight $\omega_{s,s_a}(x)$ be defined as:

$$\omega_{s,s_a}(x) = \frac{f_{X(s)}(x)}{f_{X(s_a)}(x)}. \tag{5.1}$$

That is, the weights express the magnitude of the difference in likelihood of parameter values under different data collection strategies. Then the following relation follows from Equation 4.7 using importance sampling:

$$E[g(Y)|X(s)] = \int_x g(Y(x)) f_{X(s)}(x) dx \tag{5.2}$$

$$= \int_x g(Y(x)) f_{X(s)}(x) \frac{f_{X(s_a)}(x)}{f_{X(s_a)}(x)} dx \tag{5.3}$$

$$= \int_x \omega_{s,s_a}(x) g(Y(x)) f_{X(s_a)}(x) dx \tag{5.4}$$

The above means that we can weigh the outputs $y = g(Y(x))$ obtained from the anchor strategy s_a in order to get the result for other strategies. It is critical that the weights are well defined for all possible x. In our specific setting this implies that strategy s must use the same sources as the anchor strategy s_a, but possibly with a different number of samples. In particular, we use the Central Limit Theorem as in the Basic Exhaustive Algorithm 2, and assume anchor strategy s_a is a valid strategy $(M_{i,j}^{s_a} + N_{i,j}^{s_a}) > 0$ for all $|s_a|$ pairs $(i,j) \in s_a$, where we use the superscript s_a to denote that the number of samples is that of strategy s_a. For notational convenience, let $x_{i,j}$ be a value from source $D_{i,j}$ and $[x_{i,j}]$ be an ordered sequence of $|s_a|$ input values. Then:

$$E[g(Y)|X(s_a)] = \int_{[x_{i,j}]} g(Y([x_{i,j}])) \prod_{i,j} \Phi_{i,j}^{s_a}(x_{i,j}) d[x_{i,j}] \tag{5.5}$$

where $\Phi_{i,j}^{s_a}(x_{i,j})$ is the probability density function of the Normal $\mathcal{N}(\mu_{i,j}, \frac{\sigma_{i,j}}{\sqrt{M_{i,j}^{s_a}+N_{i,j}^{s_a}}})$:

$$\Phi_{i,j}^{s_a}(x_{i,j}) = \frac{1}{\sqrt{\frac{2\pi\sigma_{i,j}^2}{M_{i,j}^{s_a}+N_{i,j}^{s_a}}}} e^{-\frac{(x_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}(M_{i,j}^{s_a}+N_{i,j}^{s_a})} \tag{5.6}$$

In Equation 5.5 we can weigh so that results for all strategies with different values $N_{i,j}^{s}$ are derived, not just for the one value of $N_{i,j}^{s_a}$ considered in the anchor strategy. So, with $N_{i,j}^{s}$ the number of samples in an alternative strategy s, we obtain:

$$\omega_{s,s_a}([x_{i,j}]) = \prod_{i,j} \frac{\Phi^{s}_{i,j}(x_{i,j})}{\Phi^{s_a}_{i,j}(x_{i,j})} \tag{5.7}$$

$$= \prod_{i,j} \sqrt{\frac{M^{(s)}_{i,j} + N^{(s)}_{i,j}}{M^{(s_a)}_{i,j} + N^{(s_a)}_{i,j}}} e^{-\frac{(x_{i,j}-\mu_{i,j})^2}{2{\sigma_{i,j}}^2}(N^{(s_a)}_{i,j}+N^{(s)}_{i,j})} \tag{5.8}$$

where we assumed that $M^{s_a} = M^{s}$ (this is a natural assumption, but the above can easily be adjusted in the exponent if this assumption is not valid). Then from Equation 4.3, Equation 5.4 and Equation 5.5 it follows that $E[g(Y)|s]$ can be derived from $E[g(Y)|s_a]$ using the Optimisation Algorithm 3. As in Basic Exhaustive Algorithm 2, we assume a total of N samples to be distributed over the data sources.

Optimisation Algorithm 3 (Importance Sampling)**.**

1. define S based on constraints (without needing results);
2. choose an anchor strategy s_a; (see Section 6.1.3)
3. do {
4. for each $D_{i,j}$ with $M^{s_a}(D_{i,j}) + N^{s_a}(D_{i,j}) > 0$ {
5. draw $x_{i,j}$ from $\mathcal{N}(\mu_{i,j}, \frac{\sigma_{i,j}}{\sqrt{M^{s_a}(D_{i,j})+N^{s_a}(D_{i,j})}})$
6. }
7. set $x = ([x_{i,j}])$;
8. for all $s \in S$ {
9. compute weights $\omega_{s,s_a}([x_{i,j}])$ as in Eq. (5.8)
10. }
11. solve $y^{s_a} = g(Y(x))$;
12. for all strategies $s \in S$ {
13. update $E[g(Y)|s]$ using y and $\omega_{s,s_a}([x_{i,j}])$;
14. (Eq.(4.3) with $\omega_{s,s_a}([x_{i,j}]) \times y$ for x_n)
15. update $Var[E[g(Y)|s]]$ using y and $\omega_{s,s_a}([x_{i,j}])$;
16. (Eq.(4.4) with y for x_n and $\omega_{s,s_a}([x_{i,j}])$ as weight
17. for each term within sum)

```
18.  }
19.  }
20.  until for all s, Var[E[g(Y)|s]] accurate
21.          (as in Section 5.4)
22.  filter S based on any constraints requiring Var[E[g(Y)|s]]
23.  select s that minimises Var[E[g(Y)|s]];
```

Comparing the Basic Algorithm with the Importance Sampling Algorithm, we see that the Importance Sampling Algorithm replaces the outer For loop in Optimisation Algorithm 1 with an inner loop (lines 8 & 12) that utilises the importance sampling equations. This should create a considerable speed up, since it implies that the original algorithm needs to be run for one strategy only (namely the anchor strategy, lines 4 & 11). However, the precise efficiency gain depends on the accuracy obtained using the importance sampling equations, which is determined by the stopping criterion. We discuss the stopping criteria in Section 5.4.

5.3.1 Choosing an Anchor Strategy

Line 2 in this algorithm requires selecting or configuring of an *anchor strategy*. When using importance sampling it is recommended that the initial distribution (the distribution that is sampled and computed directly, the numerator in Equation 5.1) should have a thicker tail. This enables a larger range of values to be sampled and used as inputs. A good anchor strategy is therefore a collection of sample sizes for data sources that produce thicker tailed distributions when representing the resulting input parameter uncertainty. These will be the data sources with the least certain initial data or data sources configured with the minimum sample sizes. Using this kind of anchor strategy will cover a relatively large range of values in the input parameter space, allowing it to more easily be re-purposed for a variety of strategies.

If all strategies being analysed by the algorithm use the same active data sources, it is theoretically possible for any anchor strategy to produce accurate algorithm results with a sufficient number of model experiments (number of values

generated for the input parameters and run through the model). Using narrower tailed distributions creates extremely uneven weights in the importance sampling calculation. This makes it more difficult to get an accurate result and requires increased sampling of the anchor strategy, reducing the efficiency improvement. A bad choice of anchor strategy means that a lot of model experiments are required to achieve good results, possibly even more than just completing evaluation using the original Basic Exhaustive Algorithm.

5.4 Uncertainty in the Algorithm Results (Var of Var)

Given that solving algorithms sample from representations of parameter data uncertainty, the algorithm result for $Var[g(Y)|s]$ is an uncertain estimate. Increasing the number of experiments R, increases the coverage of the input space, and in turn increases the certainty of the estimator. Rather than running for a set large number of experiments we want to determine a confidence interval for our $Var[g(Y)|s]$.

This can be used to decide if the algorithm results are sufficiently certain, or more importantly, compare the performance of different methods such as the importance sampling produced results. We calculate this by looking at the sample variance of $Var[g(Y)|s]$, (shortened to Var of Var). It is calculated by:

$$Var\ of\ Var = \frac{1}{R-1} \times \sum_{i=1}^{R} (z_i - \bar{z})^2 \tag{5.9}$$

$$z_i = (y_j - \bar{y})^2 . w_j \tag{5.10}$$

$$\bar{z} = \frac{\sum_{i=1}^{R} z_i}{R} \tag{5.11}$$

Essentially we are looking at the variance of the squared differences that go into calculating $Var[g(Y)|s]$. Where R : number of experiments for the strategy. y_i : output of experiment i of the strategy's (or anchor strategy's) results. $\bar{y}$: mean taken over y for a strategy. w_i : the weighting of the experiment output y_i, 1 in standard case (or calculated given the anchor and target strategies input distri-

butions, see Section 6.2.3). Note these i and j are used only as indices and are not related to any other i and j.

Calculating this value when using importance sampling allows us to compare the quality of results provided by different anchor strategies. It can also be used to estimate ratio for how much more computation would be needed to reach a similar level of accuracy.

5.5 Iterative Algorithm

In this section we explain a third optimisation solving algorithm that takes an iterative improvement based approach to finding the optimal strategy. The algorithm is a recreation of a method Freimer and Schruben [1] describe, with additional integration into our problem specification.

As part of Optimisation Problem 4 in Section 3.3.2.3, we introduced the idea of optimising strategies based on the results of an ANOVA method. The optimisation problem is only suitable for simulation-based solving scenarios since it constrains valid strategies to those where the output variance from parameter uncertainty $X(s)$ is no longer noticable. It enables one to take into account simulation variance more formally than when using the Basic Exhaustive Algorithm. From a optimisation solving perspective, the general idea is to start from base minimum strategy s_1 and increment per parameter data collection, until it is estimated that the evaluation results show no distinguishable variance due to $X(s)$.

When using ANOVA, a strategy s is not compared based on one overall variance value. Instead we attempt to portion the effect of the parameter uncertainty $X(s)$ to each input parameter in P. Using an F-test statistic (Equation 3.3) one produces separate results for the main effect of each input parameter e.g. p_1, and interaction effects for the effect of two or more parameters together e.g. $p_1 \times p_2$. Based upon these results we increment sample size $N(D_j)$ for one or more parameters, creating a new strategy to evaluate.

Optimisation Algorithm 4 (Iterative Algorithm)**.**

1. define s_1 based on constraints (and best source per parameter);
2. while $T(s) \leq C$ {
3. do {
4. for each p_i in s {
5. generate x_i using active data sources for i in D;
6. }
7. set $x = (\{x_i\})$;
8. do {
9. solve $y = g(Y(x))$;
10. update $E[g(Y)|s]$ using y;
11. update $Var[E[g(Y)|s]]$ using y;
12. } r times
13. } k times
14. calculate F_0 and p-values (for p_i and $p_i \times p_{i+1}$) using y;
15. if all p_i $F_0 \leq F_{0.05,k-1,rk-1}$
16. stop;
17. else
18. increment $\{N(D_j)\}$ using F_0 using criteria (see Section 5.5.1);
19. }

We start by defining s_1 the strategy to evaluate in the first interation (line 1). Freimer and Schruben [1] used a simplified problem domain where each input parameter has only one associated data source (in fact they do not actually go into detail of where the data comes from or its cost). For our purpose, one must select a single active data source per parameter. Multiple parameters can still use the same data source D_j but only one chosen source per parameter will be incremented when necessary. As before, one creates parameter values and experiments using a data uncertainty modelling method from Chapter 4 (line 5). The bootstrap resampling method is explained and applied by Freimer and Schruben

[1] but any method should work as long as it represents $X(s)$ and certainty increases with additional data collection.

Each experiment is solved by simulation r times (lines 8 to 13). This is one of the key differences from previous solving algorithms and it is required to do the ANOVA calculations. Repeated simulation can be included in the previous algorithms but it was not neccesary to produce results. It is redundant for analytical solving since the result should be identical for the same experiment.

After completing the $k \times r$ solutions, one can compute the test statistics using Equation 3.3 (line 14). It results in an F and related p-value (ANOVA p, not parameter p_i) for each input parameter and the interaction of each pair of input parameters.

5.5.1 Incrementing Criteria

Based on the F-test results, we decide which input parameters should be allocated further data collection (line 18). We call the new combination of $N(D_j)$ sample sizes strategy s_2 and this will be evaluated in the next iteration. The decision to increment a sample size is based on whether one of a parameter's effects is deemed significant. In Algorithm 4 an effect is significant if the p-value < 0.05, in which case the null hypothesis H_0 is rejected (Section 3.3.2.3). If none of the effects are significant the algorithm stops, and the current s is deemed optimal.

We use the criteria of Freimer and Schruben [1], slightly modified for our data source problem specification, to choose which data sources to increment for the next iteration. Let $\eta_j(D_j)$ be the next sample increment for D_j, and D_i^* be short hand for active data source for p_i. If any of the effects are significant apply the following rules:

for each p_i {
  if the main effect of p_i is significant
  and D_i^* has not be incremented this iteration
    set $N(D_i^*) = N(D_i^*) + \eta(D_i^*)$
}

if none of the main effects are significant
  for each interaction effect $p_i \times p_i + 1$ {
    if the interaction effect is significant
    and D_i^* has not be incremented this iteration
      set $N(D_i^*) = N(D_i^*) + \eta(D_i^*)$
    if the interaction effect is significant
    and D_{i+1}^* has not be incremented this iteration
      set $N(D_i + 1^*) = N(D_i + 1^*) + \eta(D_i + 1^*)$
  }

5.5.2 Stopping Conditions

The algorithm stops when all of the ANOVA effects are not significant (line 16). In this case, the strategy s from the final iteration is optimal. It is optimal because it meets the F-test constraint and by incrementing from minimum must minimise $T(s)$. The algorithm can also stop if, after incrementing data source sample sizes, the total cost of the new strategy $T(s)$ is greater than the available budget C. When that occurs, no optimal strategy was found.

5.5.3 Other Differences

The Iterative Algorithm is heavily dependent on the simulation solving variance. It is also important in other algorithms but its effect can sometimes be overcome with a large experiment sizes. Since the iterative algorithm decides no more data is needed based on differentiating parameter uncertainty variance from simulation variance, it is important that the certainty of the simulation solving is similar (or slightly better) during algorithm execution to what it will be during production model usage.

By increment from a base strategy, the iterative algorithm provides only a limited path through optimisation problem solution space. This both a benefit and a drawback. The minimal approach, only adding and testing what is predicted to be necessary, minimises the number of strategies that are evaluated by the algo-

rithm. It provides a single approximate solution for the optimal strategy, or if the budget constraint is reached no optimal solution. While this dramatically reduces model computation versus the BEA (1), we do not learn much about the other non-optimal strategies and the solution space, beyond what can be discovered from the strategy iteration history. The non-uniform costs, sample incrementing, and other factors not present in [1], means that there could possibly be more than one optimal strategy and the result is an approximation.

The Iterative Algorithm also requires the user to configure the base strategy s_1 for the first iteration. In the base strategy s_1, each input parameter p_i needs a single active data source D_i^* which will possibly be incremented at the end of each iteration. Selecting these removes the choice of data source from the problem. To decide which data source one needs some understanding of which data source will be both best for the first iterations but also remain the best choice after all the incrementing. For most problems this will be quite simple, but when there are many available data sources per parameter and complex cost functions, what is the best choice initially may not be after greatly incrementing the $N(D_j)$ sample sizes.

A more *extended algorithm* could include this data source selection problem within the solving algorithm by evaluating *multiple base strategies*, one for each combination of input-data source pairings. This could be achieved by surrounding the current algorithm in another loop over each base strategy instead of a single s_1. It would increase the required computation but it would still be more efficient than BEA as one would not be evaluating every possible sample size. Iterating and incrementing from multiple base strategies produces multiple paths through the solution space, which can result in more than one valid selection (set of strategies S in Optimisation Problem 4). The original *minimise cost* objective function would then be needed to decide the optimal strategy (or strategies) from S. This algorithm is not explored further and left as possible future work.

5.6 Exploring the Strategy Space More Efficiently

If the model has many input parameters, many data sources, or highly granular data source options, the optimisation problem can have a very large strategy solution space. In this section we further discuss why this problem may occur and present options for making the problem solving practicable. The problem can be approached from two points of view, we can reduce the amount of potential strategies (Section 5.6.1) or we can reduce the amount of strategies we evaluate to find a result (Section 5.6.2).

We define a strategy's analysis or evaluation cost as the computation of model solutions needed to evaluate a strategy, and optimisation problem complexity as the total number of strategies that need to be analysed as part of solving the optimisation problem.

5.6.1 Preparation by Screening

As discussed in Section 3.3.3, the complexity of the strategy solution space is affected by the number of input parameters $|P|$, the number of data sources per input, and the number of sample size levels within a data source. Reducing any of these three reduces the number of strategy configurations to test with the model in the Basic Exhaustive Algorithm. When using the Basic Exhaustive Algorithm with importance sampling only the first two reduce strategy evaluations with the model.

The data collection optimisation is most important and beneficial for significant *input parameters*, those that the model is at least slightly sensitive to. While less important for low complexity problems, when problem complexity becomes an issue the first and most obvious reduction is to remove any parameters that are known to be or detected as insensitive. This parameter screening (or factor screening as it is commonly known) can be achieved by using a simpler sensitivity analysis method or a dedicated method [11]. It is important that the cost of screening should be well below the cost of evaluating the strategies with the insensitive parameters, or their removal provides little benefit beyond simplified

results. The excluded insensitive parameters are likely to already have sufficient data from the existing data but if necessary one could allocate a minimal amount of additional collection in all strategies.

The second factor is *the number of data source choices*. Each new alternative choice of data source brings a whole new set of strategies to compare. For some data sources it is sometimes possible to tell before testing whether the data source is very unlikely to be selected and add nothing particularly useful to the optimisation results. For example, if a data source provides low quality data, is not cheap, and will not be needed when another preferred data source reaches its maximum sample size, it could be suitable for removal. Collection choices like that should be avoided when specify the problem scenario but they can sometimes be naïvely included for completeness. If the user can interpret signs like these with certainty, the data source could be excluded from the available options for a single input parameter or all input parameters. The exclusion from either will reduce the optimisation problem complexity.

Reducing the optimisation problem complexity resulting from *sample sizes* will be covered further in the following sections. From a pre-analysis screening perspective, we re-iterate that each data source can specify a minimum sample size $N_{min}(D_j)$, maximum sample size $N_{max}(D_j)$, and a sample increment $\eta(D_j)$ (Section 3.2.2). Restrictive specification of these values, in line with the problem domain, provides a good starting point for limiting the overall complexity. When these data source variables are flexible and optimisation problem complexity is a concern, it is best to be restrictive with their initial specification. The initial results can then be used to decide whether it is necessary to expand the solution space as discussed in Section 5.6.6.

5.6.2 Partitioning & Searching the Space

The Basic Exhaustive Algorithm can be costly because it evaluates every strategy. The opposite alternative is the Iterative Algorithm (Section 5.5) but this requires a starting strategy and produces the bare minimum results, you cannot compare the effect of changes in strategies outside the optimal. The importance

sampling modification (Section 5.3) can reduce the required model computation in the strategy evaluation cost but it still takes longer to compute as the optimisation problem complexity increases, especially with many data source choices.

To reduce the effect of the optimisation problem complexity on the solving algorithms, one can partition the solution space and use more intelligent searching. There are many methods for efficiently searching an optimisation space, here we suggest two approaches for the problem that we call *moving window* and *iterative expansion*.

5.6.3 Moving Window: Batch-based Solving

The optimisation problems in Section 3.3.2 has two independent objective functions: minimise variance and minimise cost. One can therefore approach the strategy space from two directions depending upon the objective. As before, let S be all currently valid strategies within budget (total cost $T(s) \leq C$). As one will see in the following sections, one can scan through S ordered by $T(s)$ evaluating strategies and make an informed decision on whether the optimal strategies has been found. Rather than looking at the entire S space, we look at a moving window on S. While described in terms of standard model-based strategy evaluation, these algorithms can also be used with importance sampling.

For generic and parallelising the execution it is easiest to consider and complete strategies by this method in batches. These batches can have a fixed size for simplicity and predictability, or variable size for more complex batching. A variable size would be used if evaluating strategies within defined cost ranges.

5.6.4 Cheapest-First Search Algorithm

First consider the *minimise cost objective*, this is an example of an objective that can be used to inform the search prior to calculating all the required results. In this case, evaluating the constraints needs the results for each strategy rather than the objective function. The total collection cost of each strategy can be calculated before analysing any strategies with the model. Therefore, the optimisation solving

algorithm does not need check all of S. Since the total cost is known, we can order the strategies by total cost $T(s)$ and then start testing the cheapest strategies first. The algorithm can stop when an optimal strategy is found, one that meets the remaining constraints including the variance constraint, since no subsequent strategy can better meet the objective function. The number of strategies that need to be evaluated is now between 1 and original complete set $|S|$.

Optimisation Algorithm 5 (Cheapest-First Search Algorithm)**.**

1. define S based on constraints (without needing results);
2. sort S by **ascending** total cost $\sum_{j=1}^{|D|} C_j(N(D_j))$;
3. for each $s \in S$ {
4. do {
5. for each p_i {
6. generate x_i using active data sources for i in D
7. }
8. set $x = (\{x_i\})$;
9. solve $y = g(Y(x))$;
10. update $E[g(Y)|s]$ using y;
11. (Eq.(4.3) with y for x_n)
12. update $Var[E[g(Y)|s]]$ using y;
13. (Eq.(4.4) with y for x_n)
14. }
15. until $Var[E[g(Y)|s]]$ accurate (Section 5.4)
16. if **stopping condition** true ($Var[E[g(Y)|s]] \leq V$)
17. select s and stop;
18. }

The main differences from BEA 1 are the pre-sorting of strategies (line 2) and the stopping condition (lines 16 & 17), which allows escaping from the loop without evaluating $Var[g(Y)|s]$ for all s.

5.6.4.1 Stopping Condition

The stopping condition can simply evaluate if any strategy in the latest batch has achieved the required level of variance $Var[g(Y)|s] \leq V$. For completeness, if an optimal strategy is found the algorithm should continue until strategies at the current cost level have been evaluated. There may be multiple strategies with same total cost, which are equally optimal under the minimise cost objective alone.

5.6.5 Most-Expensive-First Search Algorithm

The *minimise variance objective* is an example of an objective that depends entirely upon the analysis results, only the constraints can be evaluated prior to assessing strategies. In these conditions algorithms that do not assess every strategy only provide approximate solutions. When required we can still make assumptions based on the objective to inform the searching. We now consider the other direction, starting the analysis with the most expensive strategies first. Given that spending increases data certainty, it is reasonable to assume that the optimal strategy for the minimise variance objective will involve spending the entire collection budget or somewhere close to the data source maximums. To reduce the optimisation problem complexity with this approach, one evaluates strategies in batches, ordered by total cost starting with the most expensive first, and stop by checking when the strategy results are no longer able to or likely to improve.

Optimisation Algorithm 6 (Most-Expensive-First Search Algorithm)**.**

1. define S based on constraints (without needing results);
2. sort S **descending** by total cost $\sum_{j=1}^{|D|} C_j(N(D_j))$;
3. for each $s \in S$ {
4. ...
5. if **stopping conditions** true (see Section 5.6.5.1)
6. select s that minimises $Var[E[g(Y)|s]]$ and stop;
7. }

The important differences from BEA 1 are the pre-sorting of S by descending total cost $T(s)$ (line 2) and the stopping condition that allows the escaping of the loop (line 5) without evaluating $Var[g(Y)|s]$ for all S. This algorithm differs from Cheapest-First-Search as there are multiple conditions that need to be met to allow stopping because the solution is uncertain.

5.6.5.1 Stopping Conditions

The stopping condition is more difficult for this algorithm than Cheapest-First Search since we do not know for certain that no unchecked (cheaper) strategy provides a better result for the objective function. To save on computation the algorithm must intelligently predict whether all the optimal strategies have been evaluated. We decide the stopping condition based on three factors:

1. **Have all the most costly strategies within budget been evaluated?** Under the assumption that one of these strategies is likely to be optimal, one must always evaluate those strategies. One should also evaluate those within a set range from the budget since, due to collection restrictions, it may not be possible to completely use the budget exactly. The scanning backwards from the mostly costly strategies should at least try to include the maximum extremes of all data sources, regardless of their distance from the budget limit.

2. **If the optimal strategy is within the last batch evaluated, one must evaluate another batch.** We want to make sure the algorithm covers a little more than the minimum required the strategy space for some redundancy.

3. **Are the variance results appearing to diverge?** After the algorithm has covered the extremes of the strategy solution space, the results in newly evaluated batches of strategies are likely to be less and less optimal from increasing variance. Classifying this diverging variance is key to this algorithm being both accurate and efficient. One wants to be certain the approximate solving result is good, while also saving computation by evaluating limited amount of strategies.

5.6.6 Iterative Expansion Algorithm

Instead of checking batches of strategies starting from the extremes, the optimisation solving algorithm could sample different areas of the solution space to find areas worth exploring further. The first results probe the space before additional more granular exploration tries to find the optimal.

To complete this approach one would first reduce the problem space by limiting data sources to large sample size increments $\eta(D_j)$. Based on the results of the problem with decreased complexity, one could identify areas of the strategy space that have produced promising results (such as the low variance for minimise variance objective). In solution areas still under consideration, each subsequent algorithm iteration decreases the sample size increments closer to the original optimisation problem specification. Strategies not already evaluated in these areas are analysed.

The technique has similarities with some tree and graph searching algorithms in that we wish to review some results for each possible parameter-data source combinations, before expanding down the path into more granular configuration options based on promising results.

5.6.6.1 Stopping Conditions

The algorithm stops with an approximate solution if an optimal strategy is found after expanding and analysing all the promising areas of the solution space S. If an optimal result is not found, the algorithm stops assuming no optimal strategy exists, or previously abandoned areas must be re-considered.

It is possible to integrate ideas from the previous section into this approach. One could use the assumptions about total collection cost $T(s)$ to inform where in the solution space to sample and expand first, dividing and sampling the space unevenly. For example for minimise cost, the initial probing of the strategy space could be much closer together (less simplified sample size options) for low cost strategies and more spread out (larger sample size increments) for high cost strategies. This is a little bit like the areas have already been expanded based

on prior assumptions and knowledge rather than analysis results.

5.7 Discussion of Assumptions and Limitations

This section discusses assumptions made in optimisation solving algorithms and any other limitations of their usage.

The Basic Exhaustive Algorithm can quickly become too expensive with many input parameters, many strategies in S, or models that take significant time to solve. It remains important in understanding the approach and the other algorithms are based upon it. We attempt to make solving more efficient using a number of alternative algorithms, some more specialised to specific problems.

One can use importance sampling to greatly improve efficiency by re-using model solutions over many strategies. Its performance is based on the quality and coverage of the anchor strategy result, so in most situations some of the computation saved from using importance sampling should be used to increase R the number of experiments used in a strategy evaluation. The data source choices per parameter can affect the efficiency of importance sampling. If the distributions representing parameter uncertainty greatly differ between the target strategy and the anchor strategy (due to differing means etc.), it may be necessary to evaluate and use results from multiple anchor strategies.

Incomplete and approximate solving algorithms (Section 5.5 and Section 5.6) reduce the number of strategies evaluated to find and select a solution. The Iterative algorithm assumes simulation is used for model solving, which is highly probable for our target model area but not certain. Most-Expensive-First Search and Cheapest-First Search assume a general pattern of certainty increases with $T(s)$, which is sensible assumption but fluctuations need to be taken into account and are used to inform the stopping conditions.

Restrictions on the solution space can majorily reduce execution time but one must also balance this with flexibility. Where practical, the user may want information about the choices available rather than a single solution. This may include some flexing of the constraints the user was not previously considering. Minimal

solving like the Iterative algorithm cannot provide this.

It must be noted that most optimisation solving algorithms that do not evaluate every strategy have the potential to miss the true optimal strategy. Mitigating the chance of this occurring relies upon:

The accuracy of algorithm results must be reliable when previous results are used internally to inform searching for optimal strategies.

Minimising the uncertainty of model solutions which can lead to inaccurate algorithm results.

The validity of the searching algorithm assumptions. The assumptions need to remain valid and when possible be challenged.

Analysing strategies slightly beyond what it is necessary to find an optimal result unless certain no other strategy can be better, an algorithm should not always stop as soon as possible. It should continue along all likely avenues and continue with some redundancy where practical, in case the algorithm accuracy has caused an anomaly. This might entail expanding slightly over the border of a promising area.

Given the independence of strategy evaluation, it is important to consider parallelising the implementation, and using distributed computing where appropriate, to greatly speed up: model solving, strategy evaluation, and overall algorithm solving. Most looping over a strategies in the solving algorithms could be separated into independent jobs. The main single-threaded areas are the final selection and the testing of stopping conditions. The bottleneck of stopping conditions can be reduced with large batch sizes. Parallelism is less beneficial to the iterative algorithm. It could evaluate experiments in parallel but in its normal form cannot parallelise strategy evaluation, as only one strategy is used at time.

5.8 Summary

In this chapter we explained six algorithms for solving the data collection optimisation problems in Chapter 3. The Basic Exhaustive Algorithm evaluates all valid strategies using a method from Chapter 4. Basic with Importance Sampling Algorithm allows for the reuse of existing results to reduce the number of model executions. Iterative Algorithm provides an alternative minimal perspective for solving, using incremental improvement rather than testing all strategies, and ANOVA to assess if the uncertainty has been reduced sufficiently. Section 5.6 describes alternative algorithms that use informed searching to attempt to find the optimal solution without evaluating every possible strategy. Appendix A explains a MATLAB implementation of these solving algorithms and the methods discussed in Chapter 4. The next chapter demonstrates the solving algorithms with multiple examples.

CHAPTER 6

Evaluation

The previous chapter presented algorithms for solving the proposed data collection optimisation problem. The algorithms attempt to find the optimal strategy when collecting additional data for a model under some set scenario constraints. This chapter demonstrates these solutions and their results using multiple different examples with a MATLAB implementation and two models: an M/M/1 queue model and a business workflow PRISM model. The M/M/1 model is a simpler example well covered and often used in existing modelling and simulation literature. This allows for easier analytic solving when needed and a more complete understanding of the expected behaviour. The PRISM model provides a more realistic but also more specialised example. Further details on the implementation can be found in Appendix A.

Section 6.1 explains the M/M/1 model and the data collection scenario, before presenting results from all major methods using the model. Section 6.2 explains the business workflow PRISM model with a data collection scenario. This is tested with a selection of solving algorithms and options, informed by the results of Section 6.1. Section 6.3 discusses shared aspects of the results in more detail.

6.1 M/M/1 Queue Examples

In this section we set up and solve optimisation problems for an M/M/1 queue model. Different input and solving methods are used over multiple examples and the results are discussed.

6.1.1 Introduction

Here we re-use a version of the recurring example:

> **Example.** A single till in the supermarket is modelled using an M/M/1 queue. Inter-arrival time is exponentially distributed with rate λ and the service time is exponentially distributed with rate μ. This implies that there are two input parameters, parameter p_1 is the rate of the inter-arrival time distribution and parameter p_2 is the rate of the service time distribution. The performance measure of interest $E[Y]$ is the mean time a customer spends in the queue $E[W]$.

Note that value for the input parameters must obey $p_1 < p_2$ (in terms of values of these parameters), otherwise the queue length is infinite. In our input value generation, we ignore cases where $p_1 \geq p_2$, filtering and regenerating as discussed in Section 4.8.

A Java implementation was used for the M/M/1 model when solving by simulation. Closed-form solving was completed in MATLAB using the known formula [5]:

$$E[W] = \frac{\rho E[S]}{1-\rho} = \frac{\lambda/\mu \times 1/\mu}{1-\lambda/\mu} \tag{6.1}$$

$$E[S] = \frac{1}{\mu} \tag{6.2}$$

Using analytic solving allows us to more quickly and easily understand the effect of parameter uncertainty by removing simulation uncertainty. Not all models can

be solved analytically so we also include simulation to demonstrate the difference in algorithm computation.

Data collection scenario (1) has two sources D_1 and D_2. Let D_1 provide data for p_1 (λ) and D_2 for p_2 (μ), resulting in data source-input mapping:

$$d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Each has a set of existing data information for the respective input parameters, $D_1 : \{\hat{\mu}_1 = 1; \hat{\sigma}_1^2 = 0.85; M(D_1) = 30; c_1 = 1\}, D_2 : \{\hat{\mu}_2 = 0.5; \hat{\sigma}_2^2 = 0.3; M(D_2) = 30; c_2 = 1\}$.

Let us begin with a restricted example to demonstrate the syntax/IO using Scenario 1. If we first consider three extreme strategies when the budget is limited to 200 samples:

Table 6.1: M/M/1 Simple Strategies Input

s Strategy	lambda	mu	Total Cost
1	[1 0]	[2 200]	200
2	[1 100]	[2 100]	200
3	[1 200]	[2 0]	200

Table 6.1 shows the algorithm input information of three strategies s_1, s_2, and s_3. Each parameter p_i has a column which shows the parameter name if specified. The column contains the active data sources for p_i in the form of $[j\, N(D_j)]$, where j is the data source index and $N(D_j)$ is the number of new samples allocated to data source D_j. This column can also show $M(D_j)$ the number of existing samples, $M(D_j) + N(D_j)$ the total number of samples, the collection cost for each data source $C_j(N(D_j))$, or a combination of the four. It will be $N(D_j)$ unless stated. The final column is the total collection cost for the strategy $T(s)$.

By executing the Basic Exhaustive Algorithm (BEA, Section 5.2) we get results for each strategy. 500 experiments were generated using CLT Normal distribution-based sampling (Section 4.3), with 500 values for each parameter p_i, and 1 model solution (repetition) per experiment. This results in 500 model solutions for each

strategy. In this case the model was solved analytically. Table 6.2 shows the strategy results summary for the algorithm.

Table 6.2: M/M/1 Simple Strategies Results

s	Total Cost	Values per p_i	Experiments	Exp. Repetitions	$E[g(Y)]$ (mean)	$Var[g(Y)\|s]$	Var Of Var
1	200	500	500	1	0.53129	0.039413	0.008385
2	200	500	500	1	0.52857	0.028865	0.004461
3	200	500	500	1	0.5626	0.083175	0.054164

The first column in Table 6.2 is the strategy index, an identifier that maps strategies between tables within the same example. The second column is $T(s)$ the total collection cost, the same as in Table 6.1 but presented for side by side comparison of results against cost. The next three columns describe the experiments generated as part of modelling parameter uncertainty (see Chapter 4). Values per p_i : the number of values generated for parameter p_i, the number of model experiments created from these values, and the number of times each single experiment was solved (repetitions).

The last three columns are the analysis results for each strategy: $E[g(Y)]$ the mean of performance measure results, $Var[g(Y)|s]$ the variance given the strategy (see Equation 3.1), and the Var of Var (see Section 5.4). The mean and variance are taken over all of the strategy's computed model solutions (for the chosen output). The variance $Var[g(Y)]$ is the primary output of interest for strategy evaluation. We aim to minimise this variance (when using the minimise variance objective) or keep it below a set amount (when using the minimise cost objective). The Var of Var provides a estimate for the confidence in the strategy result. It essentially calculates the variance of the squared differences that go into calculating $Var[g(Y)|s]$. A relatively low Var of Var represents a more certain estimate for $Var[g(Y)|s]$.

If we apply the minimising variance objective, the optimal strategy from these three options in Table 6.2 is s_2 Strategy 2, which is sharing the 200 samples evenly between both input parameters $N(D_1) = 100, N(D_2) = 100$. This will be fully

explored in the next section.

6.1.2 BEA Examples

Now let us expand upon on the example in the previous section to demonstrate the Basic Exhaustive Algorithm results with a larger strategy solution space. Using data from Scenario 1, let the budget C = 500 and sample increment for both sources be 25 samples. If we include strategies where $N(D_j) = 0$, $|S| = 231$, there are 231 generated strategies to evaluate. Given the size of the strategy solution space we do not include the complete table, only portions of the results.

Once again using CLT Normal distribution sampling, let k be the number of values per parameter and R the number of experiments per strategy both $= 1000$. If we first consider *minimising variance*, Table 6.3 and Table 6.4 show the ten strategies sorted by $Var[g(Y)|s]$. We can immediately see the top strategies involve spending all or most of available collection budget. The algorithm results suggest the optimal strategy shares the available budget between both parameters but not evenly. There is an obvious preference for p_2 (μ).

Table 6.3: M/M/1 BEA Top Strategies by Smallest $Var[g(Y)|s]$ (Inputs)

s Strategy	lambda	mu	Total Cost
153	[1 200]	[2 300]	500
110	[1 125]	[2 350]	475
124	[1 150]	[2 300]	450
186	[1 275]	[2 225]	500
125	[1 150]	[2 325]	475
126	[1 150]	[2 350]	500
164	[1 225]	[2 250]	475
77	[1 75]	[2 400]	475
139	[1 175]	[2 300]	475
111	[1 125]	[2 375]	500

Table 6.4: M/M/1 BEA Top Strategies by Smallest $Var[g(Y)|s]$ (Results)

s	Total Cost	Values per p_i	Experiments	Exp. Repetitions	$E[g(Y)]$ (mean)	$\boldsymbol{Var[g(Y)\|s]}$	Var Of Var
153	500	1000	1000	1	0.51612	0.012153	0.000337
110	475	1000	1000	1	0.51199	0.012479	0.000460
124	450	1000	1000	1	0.51649	0.012955	0.000424
186	500	1000	1000	1	0.51858	0.012968	0.000323
125	475	1000	1000	1	0.5144	0.013077	0.000661
126	500	1000	1000	1	0.51983	0.013269	0.000750
164	475	1000	1000	1	0.51396	0.013388	0.000511
77	475	1000	1000	1	0.51274	0.013544	0.000691
139	475	1000	1000	1	0.51881	0.013645	0.000601
111	500	1000	1000	1	0.51383	0.013674	0.000678

We can look further into the behaviour of maximum spending by comparing all those strategies. Figure 6.1 shows only s where $T(s) = 500$. By ordering strategies by $N(D_1)$ one can see the effect of toggling complete collection budget from D_2 to D_1.

If we now consider minimising total collection cost and constrain the variance requirement at an arbitrary $V = 0.1$. Valid strategies $|S| = 221$. Table 6.5 shows (the first 10) strategies sorted by ascending total cost.

In Table 6.6 the optimal strategies require only $T(s) = 50$. When using only the minimise total cost objective, both s_3 and s_{23} are equally optimal. If one also considers minimise $Var[g(Y)|s]$ as a secondary objective, s_3 would be recommended alone. Looking at Table 6.5 we can see s_3, involves $N(D_1) = 0$ and $N(D_2) = 50$. Parameter p_2 remains slightly preferred in these results but new samples are allocated to both parameters when sufficient budget is available.

It is difficult to visually present all strategies at once due to the many dimensions that make up a strategy, even with only two parameters. One can plot high-level strategy variables such as total cost. Figure 6.2 shows 230 of the original S (231 strategies) comparing total collection cost $T(s)$ against the resulting estimate for $Var[g(Y)|s]$. One can see a decreasing downward trend as spending

Table 6.5: M/M/1 BEA Top Strategies Sorted by Cost, where $V \leq 0.1$ (Inputs)

s Strategy	lambda	mu	**Total Cost**
3	[1 0]	[2 50]	50
23	[1 25]	[2 25]	50
4	[1 0]	[2 75]	75
24	[1 25]	[2 50]	75
43	[1 50]	[2 25]	75
61	[1 75]	[2 0]	75
5	[1 0]	[2 100]	100
25	[1 25]	[2 75]	100
44	[1 50]	[2 50]	100
62	[1 75]	[2 25]	100

Table 6.6: M/M/1 BEA Top Strategies Sorted by Cost, where $V \leq 0.1$ (Results)

s	**Total Cost**	Values per p_i	Experiments	Exp. Repetitions	$E[g(Y)]$ (mean)	$Var[g(Y)\|s]$	Var Of Var
3	50	1000	1000	1	0.58544	0.094336	0.14498
23	50	1000	1000	1	0.57749	0.09537	0.09078
4	75	1000	1000	1	0.56949	0.09107	0.2803
24	75	1000	1000	1	0.54434	0.06097	0.02482
43	75	1000	1000	1	0.56145	0.09683	0.25015
61	75	1000	1000	1	0.57405	0.097339	0.13174
5	100	1000	1000	1	0.57485	0.080825	0.11465
25	100	1000	1000	1	0.54578	0.059929	0.04377
44	100	1000	1000	1	0.55047	0.06235	0.04409
62	100	1000	1000	1	0.54957	0.06393	0.03578

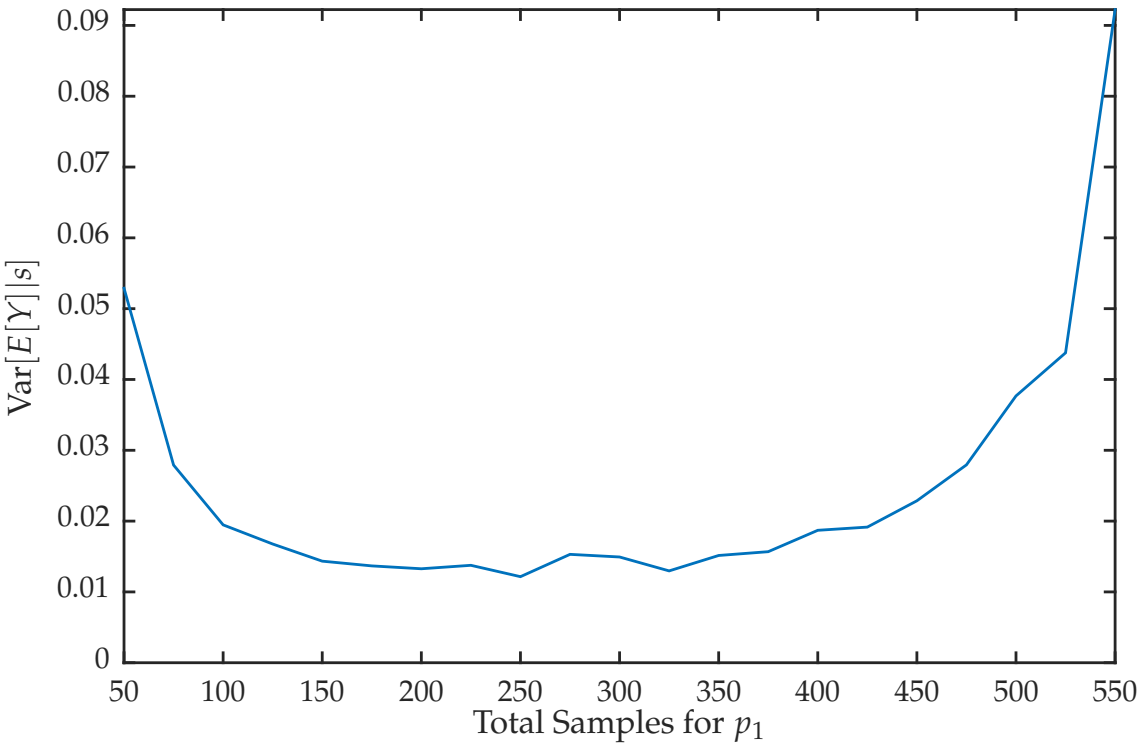


Figure 6.1: M/M/1 BEA Maximum Spend Strategies: The effect of varying samples between parameters on Variance $Var[g(Y)|s]$

on collection increases, showing diminishing returns. One can also see the spread of results at different cost levels. This is mostly likely caused by inefficiently allocating $N(D_j)$ data source samples but it can also be caused by uncertainty in the estimated results from the normal distribution sampling process.

We removed one outlier s_{16} [0, 375]; as s_{16} has a very large variance result, well beyond all others. The accuracy of the result can be questioned since the Var of Var is also very high. The values of one experiment are very close to the invalid parameter space. This is due to the random nature of the CLT normal distribution sampling and p_1 having $N(D_1) = 0$ samples. Occurrences like this can be mitigated if incorrect by either: being more conservative with the invalid range specification, increasing the number of experiments, using an experiment design approach or repeating the analysis when any Var of Var results are too large.

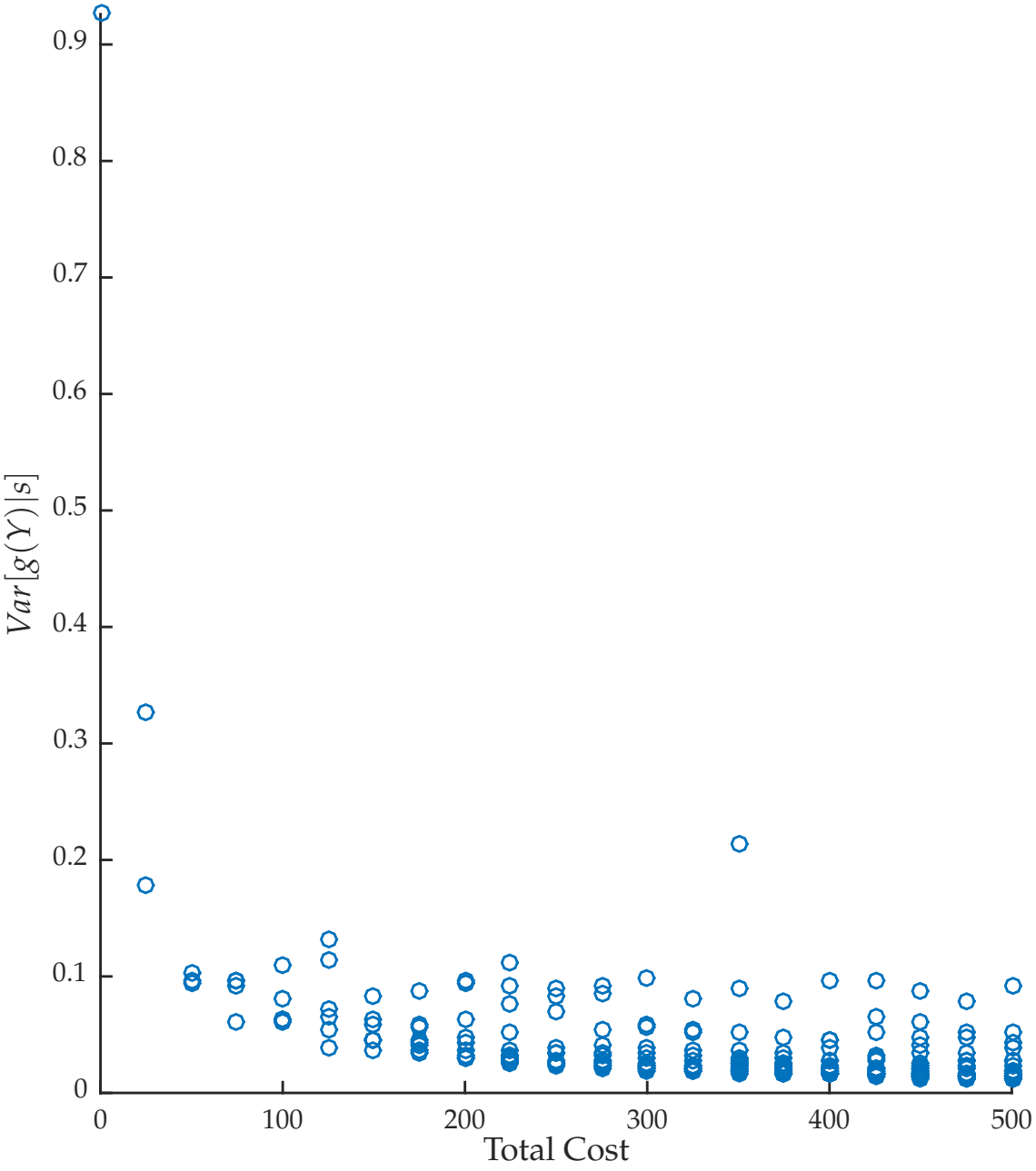


Figure 6.2: M/M/1 BEA All Strategies: The Total Cost vs Variance $Var[g(Y)|s]$

6.1.3 Importance Sampling Examples

The Basic Exhaustive Algorithm with importance sampling (Section 5.3) allows us to evaluate many strategies using the model results from one (anchor) strategy. This section will demonstrate the algorithm results, compare them against the direct evaluation of Basic Exhaustive Algorithm, and show the effect of anchor strategy selection.

The Basic Exhaustive Algorithm with importance sampling is more appropriate and efficient when the strategies being tested are using the same data sources and a good anchor strategy is chosen. The first requirement means the strategies differ only in sample sizes. This method remains valid since it could be used once for each combination of parameters and data sources. This would still provide a major reduction in model computation. Since one would need model results for one strategy at each combination of sources, rather every strategy at each combination of sources if completed via the plain Basic Exhaustive Algorithm. Here we demonstrate with strategies using only one source per parameter.

The anchor strategy is the strategy evaluated directly with the model. The results of the anchor strategy are repurposed to calculate the results of other strategies. The effect of anchor strategy choice will be explored later in the section.

For the following demonstration we define S as the maximal strategies from the previous example (those where $T(s) = C = 500$) but we also include s_0 where $N(D_1) = 0$ and $N(D_2) = 0$. We first evaluate all the strategies directly using the BEA with $R = 1000$, $k = 1000$, and $r = 1$. $|S| \times R = 22000$ model solutions used to over all the strategies.

Next, we conduct the Basic Exhaustive Algorithm with importance sampling to produce estimated results for the same strategies. In this case it re-uses the results for strategy s_0, making this the *anchor strategy*. Figure 6.3 shows the results of strategies $\{s_1, ..., s_{22}\}$ for both BEA and BEA with importance sampling. Going from left to right on the figure moves new samples from p_2 as $N(D_2)$ to p_1 as $N(D_1)$. This demonstrates that by using importance sampling we produced similar results using only the anchor strategy's 1000 model solutions instead of 1000 for each strategy.

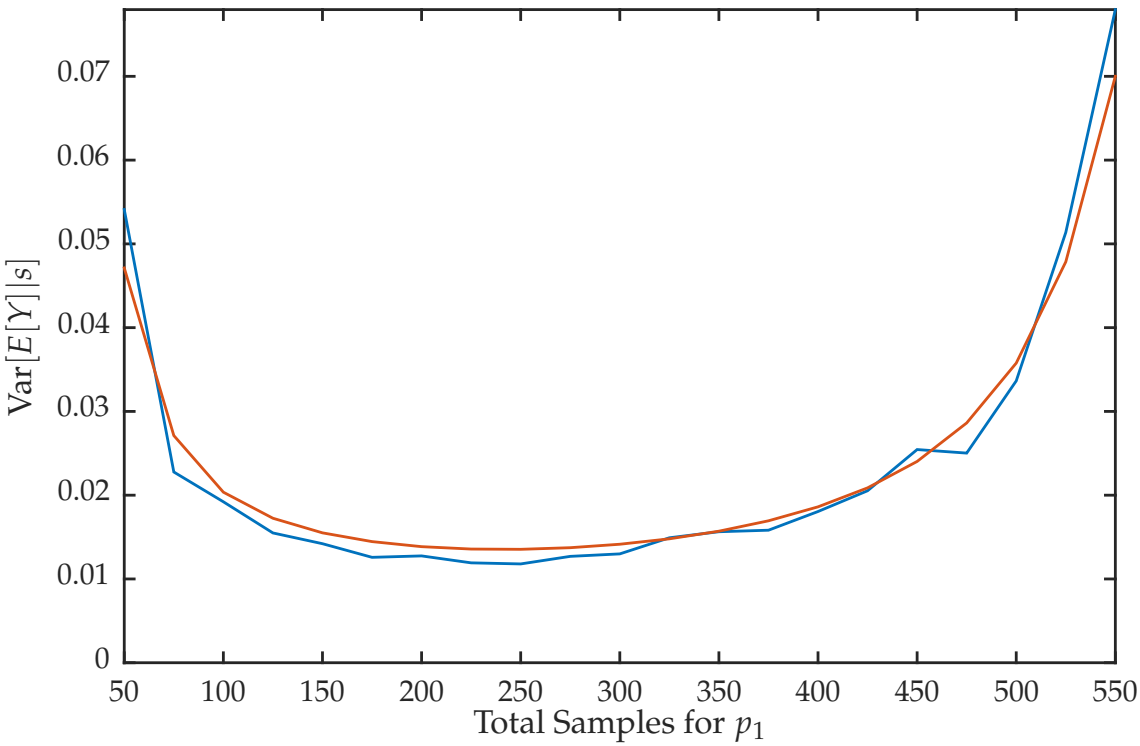


Figure 6.3: M/M/1 Comparing BEA and BEA with Importance Sampling using s_0

The small variations between the results could be caused by minor inaccuracy in the estimators of either method, and are likely caused by incomplete coverage of the parameter uncertainty. One could easily use some of the saved model computation time (from greatly reduced model experiments and solutions) to add further experiments to the anchor strategy. Under the assumption that increasing the parameter values used in evaluating the anchor strategy can improve all importance sampled strategy results. If the model solving is completed via simulation, it may also be appropriate to use algorithm efficiency saving to increase the simulation time and reduce the simulation uncertainty.

6.1.3.1 Anchor Strategies

Section 5.3.1 discussed how to choose an *anchor strategy*. The following example demonstrates the effect of different anchor strategies on the results when using the same number of experiments.

The previous example defined $s_0, ..., s_{21}$, where s_0 was the anchor strategy with

$N(D_1) = 0$ and $N(D_2) = 0$. We now show the effect of using different anchor strategies s_1, s_{11}, and s_{21}. Figure 6.3 showed s_0 with results very close to evaluating all strategies directly using the plain BEA.

Figure 6.4 shows anchor strategy $s_1 = \{N(D_1) = 0, N(D_2) = 500\}$. The results diverge on the right side of the figure, as it approaches $N(D_1) = 500$ and $N(D_2) = 0$. When solving the model for the anchor strategy strategy, the parameter data uncertainty was modelled using CLT normal distribution sampling (Section 4.3). A Normal distribution based on D_2 and $N(D_2) = 500$ is a narrower than a distribution with $N(D_2) = 0$. Most of the parameter values generated from the $N(D_2) = 500$ distribution have a much narrower range, making it more difficult to translate results via importance sampling into representing $N(D_2) = 0$. The importance sampling will involve very large weights for a few experiments to compensate. This leads to difficulties in achieving good results with the same number of experiments.

Figure 6.5 shows the opposite of s_1 in $s_{21} = \{N(D_1) = 500, N(D_2) = 0\}$. Here the results struggle to represent strategies with smaller samples for p_1 and $N(D_1)$. The same cause applies, this time for $N(D_1)$. Figure 6.6 shows the mixed case $s_{11} = \{N(D_1) = 250, N(D_2) = 250\}$. We can see how this can cause the results to diverge from the expected at both extremes. Both ends of the graph differ from the expected result.

From these results we confirm that a good anchor strategy should involve wider distributions with longer tails. The anchor strategy (or strategies) used when evaluating with importance sampling should be a strategy with low new sample sizes $N(D_j)$. Note that this may not be zero samples if that strategy is not valid or under consideration. One can use the strategy with the next smallest sample sizes, or include the $N(D_j) = 0$ strategy only to generate results for other strategies.

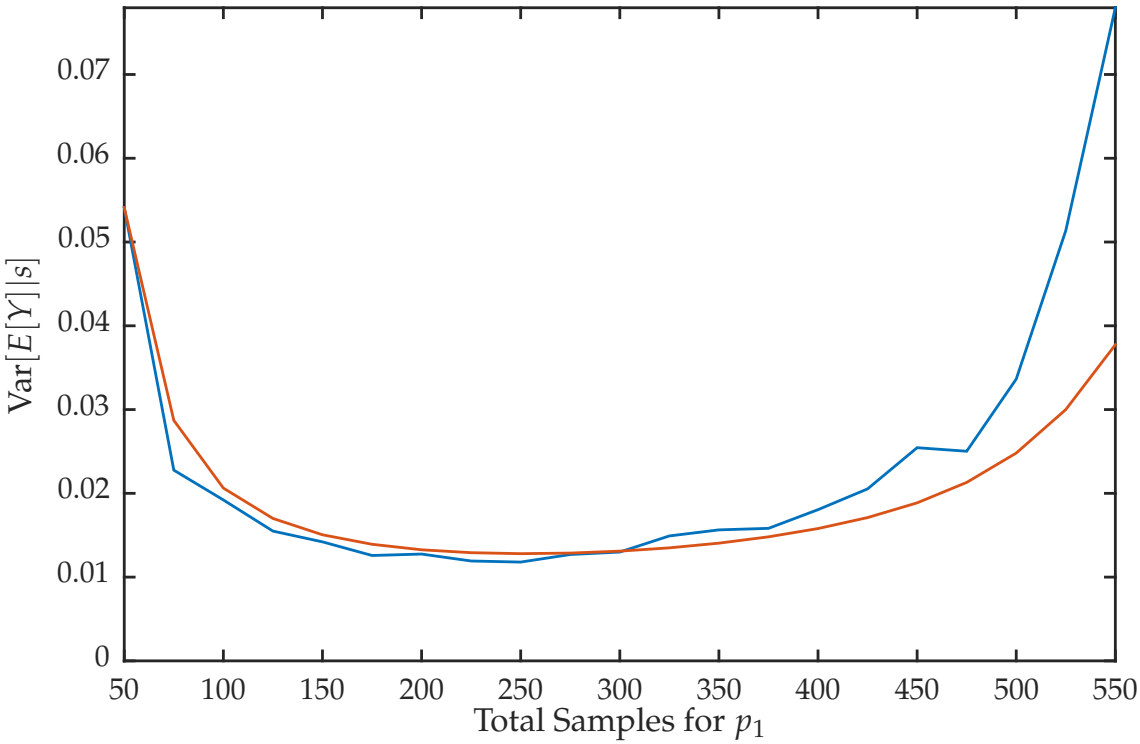


Figure 6.4: M/M/1 Comparing BEA and BEA with Importance Sampling using s_1

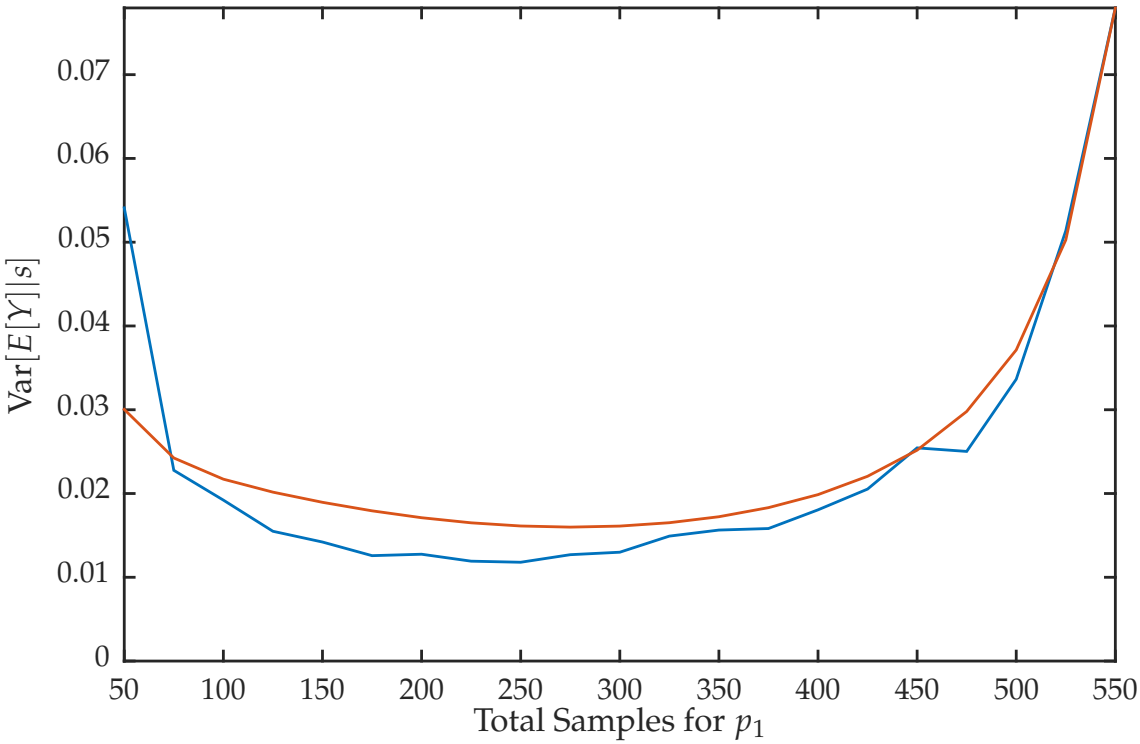


Figure 6.5: M/M/1 Comparing BEA and BEA with Importance Sampling using s_{21}

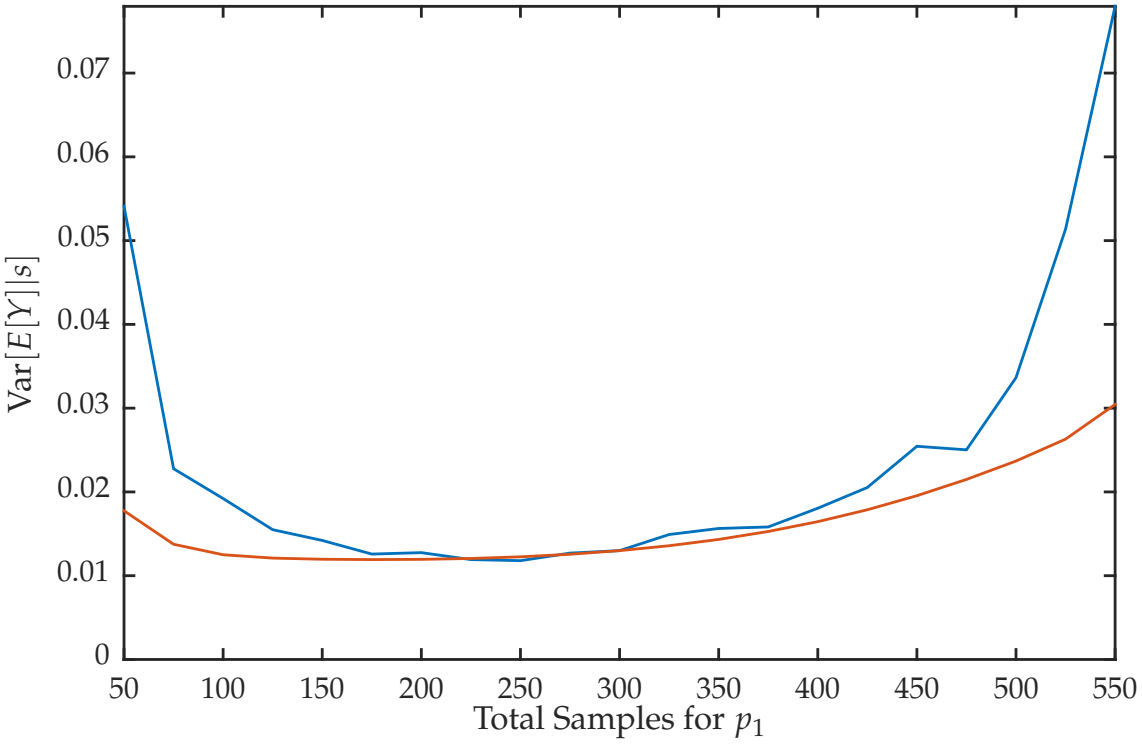


Figure 6.6: M/M/1 Comparing BEA and BEA with Importance Sampling using s_{11}

6.1.4 Iterative Algorithm

The iterative solving algorithm (Section 5.5) attempts to find the optimum strategy by iteratively adding samples, thus improving certainty, until the parameter uncertainty is no longer distinguishable from the simulation uncertainty. Rather than comparing data sources it requires a starting strategy of a selected active data source for each parameter included in the optimisation. We attempt to repeat an example from [1] using the information available, before expanding it to multiple input parameters. We continue to use the same M/M/1 model with p_1 (λ) the rate of arrival and p_2 (μ) the service rate.

The following examples use the M/M/1 model with solving by simulation and include experiment repetition not needed or used in previous examples. The total number of experiments (and therefore model executions) per strategy becomes $R = k^{|P|} \times r$, where k is the number of values (or factor levels) per parameter, and |P| is the number of parameters in the strategy. Parameter uncertainty modelling is conducted using bootstrap resampling (Section 4.6) and the $M(D_j)$ existing samples for each data source D_j.

6.1.4.1 One Input Parameter

Freimer and Schruben [1] use an M/M/1 model of a bank ATM to demonstrate their method using the ANOVA Random Effects model. While multiple parameters are discussed the results are only shown for populating a single parameter with more samples. Starting with this single parameter scenario, we attempt to allocate additional samples to improve the certainty of the mean inter-arrival time, which is used to for the rate p_1 (λ). We consider only allocating samples to D_1 for p_1. The queue has a mean customer service time of 0.9 [1], so p_2 is fixed at 1/0.9. This makes the number of parameters in the strategies $|P| = 1$.

One must configure a starting strategy s_1 for the first iteration. p_1 has one data source D_1 with $M(D_1) = 1000$ existing samples for inter arrival times. These are generated by randomly sampling 1000 times from an exponential distribution with mean 1 (the same process as [1] $\{\hat{\mu} = 1.008, \hat{\sigma} = 0.955\}$), resulting in $\{\hat{\mu}_{1,1} =$

$1.0243, \hat{\sigma}_{1,1} = 1.0004\}$. Sample size incrementing $\eta(D_1)$ is fixed at 500 samples. Since cost is not used in [1], let $c_1 = 1$. A budget $C = 10000$ is added to prevent overly long execution if a valid optimal strategy is not found. From the demonstration result in [1] we expect the final strategy to be $N(D_1) = 3 \times \eta(D_1) = 1500$, 2500 total of samples for D_1 or somewhere within one or two $\eta(D_1)$ increments of it.

The ANOVA test is setup as specified in [1]: $a = 5$ factor levels or 5 experimental values per parameter. In this case, five bootstrap resampled mean values, created using the (simulated) existing samples and the $N(D_1)$ of the current iteration. The α-value for the ANOVA test is 0.05 and we want the power of the test to be 0.95 when the ratio $\sigma_\tau^2/\sigma^2 = 0.05$. This requires $n = 240$ repetitions (r in our terms) and $\lambda = 3.6$ (test λ, not model parameter named λ). We therefore compute $R = 5^1 \times 240 = 1200$ experiments and solutions per iteration strategy. Simulation solving details are not specified, let simulation time to $t = 10000$.

Table 6.7 shows one application of the iterative algorithm to the previous scenario. The algorithm ran until the budget constraint was reached after s_{21} ($s_{21} = \{N(D_1) = 10000, ...\}$), resulting in no optimal strategies and going well beyond the expected $N(D_1) = 3 \times \eta(D_1) = 1500$.

Table 6.7: M/M/1 Iterative Algorithm Example Iterations

s	$N(D_j)$	Values per p_i	Experiments	Exp. Repetitions	$E[g(Y)]$ (mean)	$Var[g(Y)\|s]$	Var Of Var
1	0	5	5	240	6.4176	2.0769	28.361
2	500	5	5	240	5.8208	1.6598	15.567
3	1000	5	5	240	6.8282	2.3626	17.495
4	1500	5	5	240	7.0389	2.6024	50.498
5	2000	5	5	240	6.3724	1.4183	8.258
...							
19	9000	5	5	240	6.2977	1.4094	9.6782
20	9500	5	5	240	6.6152	1.5431	10.813
21	10000	5	5	240	6.6999	1.4399	8.0181

6.1.4.2 Repeated Evaluation

To further explore the solution space the same iterative algorithm execution was repeated 3 times with the budget increased to $C = 20000$. Much greater than expected to be needed for a result. The existing data for D_1 was re-generated: $\{\mu_{1,1} = 1.0257, \sigma_{1,1} = 0.9335\}$. As only one parameter is considered we can summarise the results based on under what condition the algorithm stopped (due to a chosen strategy or the budget constraint) and how many strategies (or iterations) were evaluated. The expected result should use approximately 4 strategies.

3 out of the 3 algorithm executions stopped via the constraint after 41 strategies, without finding an optimal strategy (one which met the ANOVA-based constraint). s_{41} has $N(D_1) = 20000$ new samples allocated to p_1. This clearly demonstrates a difference between these results and [1], which we assume must be related to the simulation solving.

The repeated algorithm execution was continued using different values for simulation time $t = \{1000, 5000, 10000, 20000\}$ since precise solving details were not specified in the source example. We also compare increasing the number of factor levels (values per parameter) $a = \{5, 10\}$ (discluding $t = 20000$), as this improves the coverage of the parameter uncertainty space.

Table 6.8: M/M/1 Iterative Algorithm Repeat Execution Comparison

Simulation Time t	$\boldsymbol{a = 5}$	$a = 10$
1000	s_{10}, s_{21}, s_{10}	$\varnothing, s_{28}, s_{33}$
5000	$\varnothing, \varnothing, s_{32}$	$\varnothing, \varnothing, \varnothing$
10000	$\varnothing, \varnothing, \varnothing$	$\varnothing, \varnothing, \varnothing$
20000	$\varnothing, \varnothing, \varnothing$	N/A

Table 6.8 shows the final strategy for each repetition of the algorithm at different a and t values. $\varnothing$ represents no strategy found and the budget constraint reached after $N(D_1) = 20000$. Decreasing t, and therefore simulation result certainty, causes the iterative algorithm to produce a valid strategy result more often and earlier but not particularly consistent, ranging from $N(D_1) = 4500$ to $N(D_1) = 10000$. Increasing t did not change the optimal strategy result over

three algorithm repetitions.

Given the premise that variance due to parameter uncertainty in the strategy should be indistinguishable from simulation variance (Section 3.3.2.3), it makes sense that increasing simulation variance (by decreasing simulation certainty) would increase the likelihood that a strategy is found. It also is natural to assume that decreasing simulation certainty would result in an inconsistent result for the optimal strategy. Increasing the number of factor levels a (equivalent to values per parameter) appears to result in the optimal strategy being closer to or beyond the budget constraint (and therefore not found).

6.1.5 Summary

In this section we demonstrated a selection of the solutions designed and implemented in this thesis. One can see that even with analytic solving the number of experiments is important and can greatly effect the resulting estimates for $Var[g(Y)|s]$. Stratified sampling, experimental designs, and importance sampling can allow us to stabilise these results while keeping number of experiments managable. Importance sampling must be configured correctly to provide desired performance improvement, and should be used to increase experiments used in the solution as well as reducing overall model computation. For complex models or strategy space, one should use importance sampling, use an approximate solving algorithm, or in some situations both.

6.2 PRISM Examples

In this section we present a more advanced example using a model of a business workflow with security policies.

6.2.1 Introduction

The business workflow model is a Markov Decision Point model. It is written for and solved using the PRISM model checker [80, 81]. It was provided by John Mace (Newcastle University) as part of yet to be published work (related to [92]).

It models the completion of a workflow that consists of 10 tasks being organised between 5 available users. The content of each task is left abstract but the workflow specifies security policy that restricts which users can do which tasks. The security policy can dictate which users are allowed to partake in completing certain tasks due to permissions, separation of duties, and binding of duties restrictions. For example, a permission restriction may prevent a user from being unable to access the required content needed to complete a task. With a separation of duty requirement, a user may be only allowed to work on certain tasks in the workflow (regardless of whether they are capable). A binding of duty restriction may require the same user to complete two tasks in a workflow. Complete details of the policy restrictions can be found in Appendix B.

Using the PRISM model checker the model can be solved analytically (verified) or by simulation. Its output $E[Y]$ is the maximum probability of completing the workflow given those input parameters and the internal workflow restrictions.

The **PRISM Model (PU, User-based)** associates a probability of successfully completing tasks with each user in the model. This probability is used for all tasks the user attempts. In this case it is assumed that the user's performance is consistent within a workflow and it is not dependent on the task undertaken e.g. the tasks they do are very similar. You could also assume that the security policy restrictions have externalised any apparent difference in user performance between tasks. The model takes 5 inputs parameters $\{p_1, ..., p_5\}$, identified in

the model as $\{u_1, ..., u_5\}$. Each parameter is the probability the associated user successfully completes any task undertaken.

We create a single data source for each parameter $\{D_1, ..., D_5\}$, the data source-input mapping d is therefore an identity matrix of size 5. Let the existing data for all data sources be $\{\hat{\mu} = 0.8, \hat{\sigma}^2 = 0.05, M(D_j) = 50\}$, and the per sample cost $c_j = 1$.

The scenario for examples in this section use Optimisation Problem 2B where we wish to minimise variance $Var[g(Y)|s]$. Tables of results are sorted accordingly. The budget constraint is specified separately for each example.

6.2.2 Basic Exhaustive Algorithm (Restricted)

We apply the BEA using CLT Normal distribution sampling to a restricted strategy space to test the behaviour of the results and the problem complexity of the PU model.

The PU model takes longer to analytically solve than the M/M/1 model, approximately 0.5 seconds. The PRISM software does not accept lists of experimental values, only single values or ranges, so every experiment evaluation must be a fresh and separately launched process. PRISM is integrated into MATLAB by launching the PRISM command-line application from the system and loading the exported result from a CSV file. Each command execution in MATLAB adds a small amount of extra time for shell setup and file reading. Parallel execution of experiments is used to slightly reduce both the effect of these overheards and the overall execution time.

Let the sample increment for all sources in D be 200 and the budget constraint $C = 600$. Generating all strategies within the constraints gives 56 strategies when $N(D_j) = 0$ is allowed. Let $k = 200$ and $r = 1$, then $R = 200$ experiments per strategy.

Table 6.9: PU Basic Exhaustive Algorithm Sorted by $Var[g(Y)|s]$ (Top and Bottom)

Row	p_1	p_2	p_3	p_4	p_5	Total Cost	$E[g(Y)]$ (mean)	$\boldsymbol{Var[g(Y)|s]}$	Var Of Var
1	[1 0]	[2 200]	[3 0]	[4 200]	[5 200]	600	0.11398	0.00017626	7.6913e–08
2	[1 0]	[2 200]	[3 200]	[4 200]	[5 0]	600	0.11326	0.00018298	9.2344e–08
3	[1 0]	[2 200]	[3 200]	[4 0]	[5 200]	600	0.11412	0.00018769	9.7057e–08
4	[1 0]	[2 200]	[3 0]	[4 0]	[5 400]	600	0.11522	0.00021951	9.5463e–08
5	[1 200]	[2 200]	[3 200]	[4 0]	[5 0]	600	0.11352	0.00022688	1.3709e–07
6	[1 0]	[2 200]	[3 0]	[4 0]	[5 200]	400	0.11706	0.00022832	2.6464e–07
7	[1 0]	[2 200]	[3 0]	[4 400]	[5 0]	600	0.11437	0.00023058	1.7518e–07
8	[1 200]	[2 200]	[3 0]	[4 0]	[5 200]	600	0.11515	0.00023553	1.2357e–07
9	[1 0]	[2 400]	[3 0]	[4 0]	[5 200]	600	0.11689	0.0002384	9.9486e–08
10	[1 0]	[2 200]	[3 0]	[4 200]	[5 0]	400	0.1146	0.00024006	1.671e–07
11	[1 0]	[2 400]	[3 0]	[4 200]	[5 0]	600	0.11474	0.00024258	1.1838e–07
12	[1 0]	[2 400]	[3 200]	[4 0]	[5 0]	600	0.11608	0.00024774	1.1835e–07
...									
52	[1 0]	[2 0]	[3 0]	[4 600]	[5 0]	600	0.11861	0.00047045	7.7461e–07
53	[1 0]	[2 0]	[3 0]	[4 200]	[5 0]	200	0.11991	0.00048493	6.8525e–07
54	[1 600]	[2 0]	[3 0]	[4 0]	[5 0]	600	0.11539	0.00048698	4.8468e–07
55	[1 0]	[2 0]	[3 0]	[4 0]	[5 0]	0	0.11794	0.00051235	6.648e–07
56	[1 0]	[2 200]	[3 0]	[4 0]	[5 0]	200	0.12025	0.0005157	9.0915e–07

Table 6.9 shows the top and bottom strategy results when sorted by ascending $Var[g(Y)|s]$. The top is effectively those that minimise $Var[g(Y)|s]$ (objective). There is a preference for p_2 and a smaller preference for p_5, which appears to be the most sensitive parameters. p_2 does not completely dominate since strategies allocating most (400) or all (600) samples to p_2 are not the top strategies. In general, strategies with samples allocated to only a single parameter's data source are towards the bottom of the (complete) results, suggesting that at least some spreading of the samples is more optimal. p_1 appears to be the least sensitive parameter.

The difference between the variance values is small, which makes the strategy results ordering highly susceptible small changes in the quality of the parameter uncertainty representation. This makes the number of experiments more important. It is natural to assume that the strategy with no additional samples (only existing data used for the parameter uncertainty) should be last. In these results it is second last suggesting some inaccuracy in the results. The last strategy in row 56 has a relatively large Var of Var value so it is likely to be a poor quality result rather than least optimal.

Even with analytic solving available the analysis can become costly as the number of input parameters increases. This example of 200 experiments for each of the 56 strategies (11200 experiments total) took approximately 240 minutes to compute on one quad-core Mac (parallelised where possible, used for all future times). An average of 4.3 minutes per strategy, additional strategies would be a linear increase in the total analysis time. At nearly 1.3 seconds per experiment, the model cost could be a lot worse. Given the experiment design and amount of experiments appears to be insufficient, we will use importance sampling to expand the strategy space while increasing the number of experiments used per strategy.

6.2.3 Importance Sampling

By using the Basic Exhaustive Algorithm with importance sampling, one can dramaticially reduced the required model computation. This section uses that algo-

rithm to allow more expanded strategy options. The following example uses interval-based sampling to guarantee good coverge of the anchor strategy's $X(s)$ parameter data uncertainty. Combining this with a full factorial experiment design results in an increased number of experiments per strategy compared to the previous example but a reduction in the number of different values per parameter. With $k = 5$ and 5 input parameters, $R = k^{|P|} = 5^5 = 3125$ experiments per strategy. The increase is negated by only evaluating one (anchor) strategy directly, as apposed to 56 strategies in the previous example. A total of 11200 experiments were executed in the previous example, but a different, simpler design was used.

The collection budget constraint is increased to $C = 1000$ and data source sample increments $\eta(D_j)$ remain at 200 samples. There are 252 generated strategies when completely covering the expanded problem. Thus the model computation time saved by using importance sampling is used to evaluate the anchor strategy with increased accuracy (more experiments) and evaluate more overall strategy configurations. The anchor strategy is set to $s_1 = \{...|N(D_1) = 0, N(D_2) = 0, N(D_3) = 0, N(D_4) = 0, N(D_5) = 0\}$ as recommended in Section 5.3.1 and Section 6.1.3.

Table 6.10: PU Importance sampling: Sorted by $Var[g(Y)|s]$ (Top and Bottom)

s	p_1	p_2	p_3	p_4	p_5	Total Cost	$E[g(Y)]$ (mean)	$\boldsymbol{Var[g(Y)|s]}$	Var Of Var
189	[1 200]	[2 400]	[3 200]	[4 0]	[5 200]	1000	0.11267	8.3298e–05	1.0208e–07
86	[1 0]	[2 200]	[3 400]	[4 200]	[5 200]	1000	0.11169	9.2348e–05	9.15e–08
80	[1 0]	[2 200]	[3 200]	[4 400]	[5 200]	1000	0.10848	9.4265e–05	1.1092e–07
174	[1 200]	[2 200]	[3 200]	[4 0]	[5 400]	1000	0.11325	9.4752e–05	1.0741e–07
106	[1 0]	[2 400]	[3 200]	[4 200]	[5 200]	1000	0.10804	9.6471e–05	1.3559e–07
224	[1 400]	[2 200]	[3 200]	[4 0]	[5 200]	1000	0.11349	9.6806e–05	1.2209e–07
173	[1 200]	[2 200]	[3 200]	[4 0]	[5 200]	800	0.11244	0.00010051	9.6775e–08
78	[1 0]	[2 200]	[3 200]	[4 200]	[5 400]	1000	0.10862	0.0001094	1.3147e–07
179	[1 200]	[2 200]	[3 400]	[4 0]	[5 200]	1000	0.11646	0.00011198	1.5081e–07
77	[1 0]	[2 200]	[3 200]	[4 200]	[5 200]	800	0.10785	0.00011478	1.2673e–07
...									
249	[1 800]	[2 0]	[3 0]	[4 200]	[5 0]	1000	0.14375	0.0013522	1.4555e–05
240	[1 600]	[2 0]	[3 200]	[4 200]	[5 0]	1000	0.14261	0.0013803	2.162e–05
141	[1 200]	[2 0]	[3 0]	[4 800]	[5 0]	1000	0.14252	0.0014143	1.12e–05
155	[1 200]	[2 0]	[3 400]	[4 200]	[5 0]	800	0.14344	0.0014768	1.7284e–05
160	[1 200]	[2 0]	[3 600]	[4 200]	[5 0]	1000	0.14348	0.0014923	2.0257e–05
151	[1 200]	[2 0]	[3 200]	[4 600]	[5 0]	1000	0.14213	0.0014934	2.0257e–05
215	[1 400]	[2 0]	[3 400]	[4 200]	[5 0]	1000	0.14479	0.0015892	2.7102e–05
157	[1 200]	[2 0]	[3 400]	[4 400]	[5 0]	1000	0.14433	0.0016672	2.723e–05

Table 6.10 shows the top and bottom strategy results when sorted by the objective minimise $Var[g(Y)|s]$ (calculated using importance sampling). One can immediately see that the strategies which minimise $Var[g(Y)|s]$ allocate new samples to the data sources of the assumed most sensitive parameters p_2 and p_5. This is demonstrated at both the top and bottom of the table. The overall optimal strategy when minimising $Var[g(Y)|s]$ is remains a split allocations of samples, s_{189} spreads the budget C over multiple data sources.

The bottom of Table 6.10 is a little unexpected as it is not filled with the cheapest strategies, using zero or small amounts of new samples. Instead it is strategies allocating significant sample sizes to the two parameters believed to be the least sensitive (p_1 and p_4). These strategies also have the some of highest Var of Var values, denoting relatively less certain algorithm results. It may also highlight that spending the entire collection budget does not guarantee improvement.

The complete analysis of 252 strategies using BEA with importance sampling took approximately 112 minutes. This averages out at roughly 0.4 minutes per strategy, directly evaluating the anchor strategy with the model takes a significant portion of that run time.

6.2.4 Most-Expensive-First Search Algorithm

In Section 5.6 we discussed ways of intelligently searching a large strategy solution space so that only strategies likely to be optimal are evaluated. This can be used even when evaluating strategies using importance sampling.

With the optimisation objective minimise $Var[g(Y)|s]$, one can use the Most-Expensive-First Search algorithm (Section 5.6.5) and evaluate only a subset of the strategy space to find an approximate solution to the optimisation problem.

Another bonus of the importance sampling technique is that we can re-shape the strategy solution space and still use the previous anchor strategy results (as long as the underlying data source choices and existing sample data are unchanged). If we increase the budget to $C = 1600$, the complete strategy space becomes 1287 strategies. Using the previous anchor strategy (s_1) results from

3125 experiments, we use a Most-Expensive-First Search version of BEA with importance sampling to produce an approximate solution.

Analysis stopped after 520 out of 1287 strategies, as the stopping conditions were met (see Section 5.6.5). The optimal strategy s_{402} shares the new samples between parameters p_2 to p_5. p_2 still receives the most samples in the top strategies. p_1 appears to be insensitive and should possibly have been removed from consideration before algorithm execution. This highlights the importance of factor screening (see Section 5.6.1).

Analysis took approximately 18 minutes with pre-calculated anchor strategy results. This means that a 3125 experiment, 5 input parameter anchor strategy result can be importance sampled to new strategies using approximately 2 seconds per target strategy. 520 strategies evaluated out of 1287 is close to only 40% of the strategy solution space. While not a huge time saving in this case, that would be a major time saving with a larger strategy solution space or a longer model solving time. This can be further tuned with algorithm options on batch size and stopping conditions (see Section 5.6.5).

Table 6.11: PU Most-Expensive-First Search: Sorted by $Var[g(Y)|s]$ (Top Results)

s	p_1	p_2	p_3	p_4	p_5	Total Cost	$E[g(Y)]$ (mean)	$\boldsymbol{Var[g(Y)|s]}$	Var Of Var
402	[1 0]	[2 600]	[3 200]	[4 400]	[5 400]	1600	0.11157	5.8124e–05	9.6045e–08
331	[1 0]	[2 400]	[3 200]	[4 600]	[5 400]	1600	0.11157	5.9179e–05	9.6421e–08
346	[1 0]	[2 400]	[3 400]	[4 400]	[5 400]	1600	0.1133	6.2269e–05	1.2022e–07
404	[1 0]	[2 600]	[3 200]	[4 600]	[5 200]	1600	0.11294	6.4542e–05	1.2306e–07
447	[1 0]	[2 800]	[3 200]	[4 200]	[5 400]	1600	0.11356	6.4975e–05	1.286e–07
333	[1 0]	[2 400]	[3 200]	[4 800]	[5 200]	1600	0.11242	6.6865e–05	1.0264e–07
412	[1 0]	[2 600]	[3 400]	[4 200]	[5 400]	1600	0.11488	6.701e–05	1.6721e–07
449	[1 0]	[2 800]	[3 200]	[4 400]	[5 200]	1600	0.11338	6.7064e–05	1.4024e–07
358	[1 0]	[2 400]	[3 600]	[4 400]	[5 200]	1600	0.11249	6.7125e–05	1.0753e–07
356	[1 0]	[2 400]	[3 600]	[4 200]	[5 400]	1600	0.11268	6.7594e–05	1.1258e–07
774	[1 200]	[2 600]	[3 200]	[4 0]	[5 600]	1600	0.11393	7.0961e–05	1.3885e–07
343	[1 0]	[2 400]	[3 400]	[4 200]	[5 600]	1600	0.11094	7.1615e–05	1.1151e–07
420	[1 0]	[2 600]	[3 600]	[4 200]	[5 200]	1600	0.11407	7.2242e–05	1.445e–07
984	[1 400]	[2 400]	[3 200]	[4 0]	[5 600]	1600	0.11276	7.3293e–05	1.5188e–07

6.2.5 One to Many

The following example increases the complexity of the problem description by introducing data sources that provide data for multiple parameters simultaneously. This also introduces the possibility that a parameter may have data available from more than one data source in a single strategy.

Using the same PRISM model parameters $\{p_1, ..., p_5\}$, we define a different collection of four data sources D with a data source-input mapping:

$$d = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The key differences from the previous examples are: (1) D_4 can provide data for p_3, p_4, and p_5 simultaneously. (2) if both D_3 and D_4 are active in a strategy, p_5 could use D_3, D_4, or if appropriate both.

Each data source has the same sample increment $\eta(D_j) = 200$ and the existing data is configured as: $D_j = \{\hat{\mu}_i = 0.8, \hat{\sigma_i}^2 = 0.05, M(D_j) = 50\}$. Sample cost $c_j = 1$ for all data sources except D_3, which is $c_3 = 2$. Given this data source information we can generate the base strategies (each possible combination for pairing an input parameter to one active data source). All strategies in the solution space will be based on one of these.

Table 6.12: PU One to Many Base Strategies by Input Parameter

Base Strategy	p_1	p_2	p_3	p_4	p_5
1	[1 0]	[2 0]	[3 0]	[3 0]	[3 0]
2	[1 0]	[2 0]	[3 0]	[3 0]	[3 0; 4 0]

Each cell in Table 6.12 contains the index of active data sources in the strategy with the minimum new sample size of the data source. Note in base strategy 2 while the selected data source for p_5 is D_4, data can also be provided by D_3.

The objective of this example is *minimise variance* $Var[g(Y)|s]$ subject to a max-

imum collection budget $C = 1600$. Generating all valid strategies results in 390 strategies. Using BEA with importance sampling, we evaluate all of these strategies with anchor strategy set to $s_1 = \{...|N(D_1) = 0, N(D_2) = 0, N(D_3) = 0\}$. The experiment count is increased: values per parameter $k = 6$ with a 6 interval-based full factorial design, $R = 6^5 = 7776$ experiments.

The *data combining mode* for the parameter uncertainty modelling of each input is set to *Best Estimated*. When more than one data source is active for a parameter, only data from one data source is used (the one predicted to produce data with smallest variance given the strategy information).

Table 6.13 shows the top and bottom strategies when sorted by $Var[g(Y)|s]$ (objective). Two strategies are estimated to be equally optimal s_{86} and s_{376}. With the chosen data combining mode these strategies are in fact equal as D_4 in s_{376} (existing samples only) will not be used when D_3 has a greater sample size. Since D_3 will have a narrower normal distribution representing parameter uncertainty given the data source information. The optimal strategy spreads samples across all parameters but includes a major allocation for p_1 via D_1. That allocation is slightly surprising given it appeared to be quite insensitive in previous examples. It could be partly due to the increased cost of samples for multi-parameter data source D_3 but it is more likely due to inconsistent behaviour of results between examples.

The top strategy results in Table 6.13 also include other strategies with equal $E[g(Y)]$ and $Var[g(Y)|s]$ estimators because of similar combining mode selection decisions. In some of these cases e.g. s_{140} more costly strategies with unused data collection are equally optimal. Employing minimise cost as a *secondary objective* can help migate the effect of these. While equal strategy results are possible, they are only likely to occur when importance sampling is used, and only then with a data selection mode that can result in strategies with identical parameter uncertainty distributions. If plain BEA was used for this problem description the results for these strategies would be similar but differ at least a small amount due to the different randomly generated parameter values in experiments.

Table 6.13: PU One to Many Sorted by $Var[g(Y)|s]$ (Top Results)

s	p_1	p_2	p_3	p_4	p_5	Total Cost	$E[g(Y)]$ (mean)	$\boldsymbol{Var[g(Y)|s]}$	Var Of Var
86	[1 1000]	[2 200]	[3 200]	[3 200]	[3 200]	1600	0.10985	8.3145e–05	1.1511e–07
376	[1 1000]	[2 200]	[3 200]	[3 200]	[3 200; 4 0]	1600	0.10985	8.3145e–05	1.1511e–07
359	[1 800]	[2 200]	[3 200]	[3 200]	[3 200; 4 200]	1600	0.11714	0.00011322	3.3403e–07
78	[1 800]	[2 200]	[3 200]	[3 200]	[3 200]	1400	0.11714	0.00011322	3.3403e–07
358	[1 800]	[2 200]	[3 200]	[3 200]	[3 200; 4 0]	1400	0.11714	0.00011322	3.3403e–07
140	[1 0]	[2 200]	[3 600]	[3 600]	[3 600; 4 200]	1600	0.11829	0.00013589	6.0612e–07
9	[1 0]	[2 200]	[3 600]	[3 600]	[3 600]	1400	0.11829	0.00013589	6.0612e–07
139	[1 0]	[2 200]	[3 600]	[3 600]	[3 600; 4 0]	1400	0.11829	0.00013589	6.0612e–07
76	[1 800]	[2 0]	[3 400]	[3 400]	[3 400]	1600	0.11003	0.00015365	1.0052e–06
353	[1 800]	[2 0]	[3 400]	[3 400]	[3 400; 4 0]	1600	0.11003	0.00015365	1.0052e–06
...									
197	[1 200]	[2 0]	[3 0]	[3 0]	[3 0; 4 1200]	1400	0.19977	0.011073	0.00048738
243	[1 200]	[2 600]	[3 0]	[3 0]	[3 0; 4 800]	1600	0.19824	0.011107	0.001152
147	[1 0]	[2 400]	[3 0]	[3 0]	[3 0; 4 1200]	1600	0.19955	0.011192	0.00072357
216	[1 200]	[2 200]	[3 0]	[3 0]	[3 0; 4 1000]	1400	0.20087	0.011662	0.00090593
128	[1 0]	[2 200]	[3 0]	[3 0]	[3 0; 4 1400]	1600	0.20313	0.012121	0.00068045
232	[1 200]	[2 400]	[3 0]	[3 0]	[3 0; 4 1000]	1600	0.20735	0.014255	0.0016616
198	[1 200]	[2 0]	[3 0]	[3 0]	[3 0; 4 1400]	1600	0.20993	0.014639	0.00084022
217	[1 200]	[2 200]	[3 0]	[3 0]	[3 0; 4 1200]	1600	0.21228	0.015881	0.0016901

Given the previous examples some of the strategy ordering is a little unexpected, especially since p_1 was previously thought to be insensitive and p_5 one of the more sensitive parameters. There are a notable number of strategies with relatively larger Var of Var values towards the bottom of the strategies results (maximising $Var[g(Y)|s]$), many of which have larger sample sizes for D_4 (p_5 only data source). The results suggest further exploration of trying to normalise the confidence of algorithm strategy results is required. This may involve using a variable number of experiments based upon the Var of Var confidence measure or new solving techniques.

There is also the possibility that importance sampling from the widest parameter uncertainty distribution to the narrowest struggles to maintain comparable strategy result quality (with the current experiment design). If the anchor strategy is not a real strategy under consideration, and we do not evaluate the $Var[g(Y)|s]$ for it, the anchor strategy experiments used in importance sampling do not need to be the same single source distribution for all experiments. Instead, it may better for the anchor experiments to be made up of values sampled from multiple source distributions, both wide and narrow distributions. This in turn covers both the required range and the likelihood of values in the target strategies. Analysing a strategy with importance sampling could also use only a subset of the more likely experiments values, to stabilise the importance sample weights. Further exploring this factor of importance sampling is beyond the scope of this work.

6.2.6 Summary

This section of PRISM-based examples highlights the need to consider solving complexity and to further explore the relative confidence of algorithm strategy results.

The initial results in this section appeared to highlight a need to increase the number of experimental values. While the model execution cost is relatively small, the overall time still increases rapidly when evaluating a full factorial design or a large strategy space. BEA quickly becomes long running to impractical

with limited computational resources.

The proposed algorithms do not appear to be viable for this model at granular sample sizes and may need further future work to produce useful consistent results at the demonstrated sample sizes. Differences in behaviour between examples brought further questions about the relative quality of the $Var[g(Y)|s]$ estimates. Even when using an increased number of experiments in later examples, this uneven quality appears to dominate, making general strategy ordering or patterns difficult to resolve with certainty. Further work is needed to normalise $Var[g(Y)|s]$ confidence (during or after solving) before the results can be usable in practice.

6.3 Discussion

This section provides a further discussion of the results in Section 6.1 and Section 6.2 including what can be learned from them collectively.

The results show that we are dealing with very small values for variance and other results, which makes the result very sensitive to errors from algorithm or solving inaccuracy. With large or unpredictable simulation variance it may be hard to distinguish the effect of parameter uncertainty, although one could also argue that that is a good thing. If the input, and specifically, parameter uncertainty is deemed the next best improvement to the model results, then simulation uncertainty must already be manageable.

Larger data source sample size increments make the optimisation problem both easier to deal with and easier to see results, since changes have a more noticable effect on the uncertainty between sample levels. This is why others [1] use large increments like 500 in an example. It would not necessarily scale easily to much more granular sample sizes. One could claim that this shows a problem in evaluating the algorithm results to sufficient accuracy (given very small numbers), or that we must also consider whether the parameter uncertainty modelling is representative. Either way this could be greatly improved with the use of more realistic data.

Even with small models the computation costs multiply rapidly. The importance sampling-based approach becomes necessary rather than an optional efficiency improvement. Other methods sometimes resort to metamodel simplifications in place of actual models. These are trained with some experiments to then solve for many inputs. While we have avoiding doing this directly, one could consider our importance sampling approach similar to these. When using models that take longer to solve than those presented here, anything beyond small strategy solution space will likely need to use the approximate solving algorithms.

6.4 Summary

In this chapter we presented the results from applying the optimisation solving algorithms with different options to two models: an M/M/1 queue model and PRISM based business workflow model.

Section 6.1 (M/M/1) demonstrated features of the various methods presented in the thesis. It showed the techniques working but only in an ideal, less complex scenario. Section 6.2 (PRISM) applies what was learned in the earlier sections to a less than ideal scenario. One sees the issues when applying the algorithms to more complex models, many of which are common to sensitivity and uncertainty analysis type methods. The results were then discussed as whole considering what can be learned from evaluation across different models. The next chapter concludes the thesis by summarising the contributions of the thesis and describing possible directions of future work in this area.

CHAPTER 7

Conclusion and Future Work

This thesis has looked at optimising the data collection strategy when collecting data for stochastic models and provided a number of contributions in this area. The following chapter reviews the key contributions of the thesis and the limitations.

The rest of the chapter is structured as follows: Section 7.1 highlights the contributions of this thesis, including how they link to each chapter. Section 7.2 summarises limitations in the proposed solutions. Section 7.3 discusses different directions of future work in this interesting problem area.

7.1 Summary of Contributions

This section summarises the key contributions of the thesis, and relates them to the appropriate chapter.

- **A background and literature review of data collection for stochastic and probabilistic models and related model analysis topics.**
  We provided a cross discipline review of the diverse problem area and related work. Chapter 2 explained the key topics and issues relating to the scenario, before discussing existing work that is similar to our problem or a subproblem within it. This included the essential aspects of sensitivity and uncertainty analysis. These techniques overlap with the thesis problem but are tools in the process rather than a solution to our optimisation. Similar

work in this area does not consider the expanded data collection problem, only the need for more data.

- **A problem formulation for expressing the data collection problem and a collection of optimisation problems**
  We formally defined key aspects of the data collection problem domain and presented a framework for describing it throughout the thesis. In Chapter 3 the data collection problem was broke down into a number optimisation problems, which were then expanded to include more complex aspects of the problem. We provided multiple perspectives on the problem, looking at minimising uncertainty or minimising cost, and a thorough consideration of the collection constraints not often mentioned in related work.

- **Algorithms to allow the solving of the optimisation problem, working strategy qualities through the model.**
  Multiple algorithms were developed for different approaches to solving the optimisation problem, and another based on related work [1] was integrated into our framework to encapsulate additional constraints. Each uses the model to estimate the effect of a strategy's parameter data uncertainty on the model outputs. Chapter 5 presented details of the algorithms and a discussion of their assumptions and complexity. The value of our approach is the ability to optimise the allocation of additional data samples within the problem constraints.

- **Additional optimisations for improving algorithm efficiency using importance sampling and other techniques.**
  Chapter 5 provided an importance sampling based alternative that can greatly reduce the required model computation. We added stratified sampling and experiment designs to better cover the parameter uncertainty space of a strategy with less model executions (Chapter 4).

  Reducing the required model solutions is not the only way we can improve efficiency. We also looked to reduce the numbers of strategies that need to be evaluated to find the optimal strategy. Additional algorithms were presented that intelligently search the space based upon information known

before the evaluation and the results evaluated so far. These can dramatically reduce the number of strategies evaluated, which in turn can reduce model solutions needed.

- **A prototype MATLAB tool allowing generic execution of the optimisation solving algorithms and associated functions.**
  The solving algorithms and all related methods were developed and tested in MATLAB. The MATLAB tool interfaces with external models in PRISM and Java as required. Where possible, components of the tool were split up and made model independent. For example, the models, input generation methods and objectives can be swapped out relatively easily. Appendix A provides a detailed description of the implementation.

- **A demonstration and evaluation of the optimisation implementation using multiple examples.**
  We evaluate the algorithm results in Chapter 6 using an M/M/1 model and a PRISM model. The M/M/1 model is a simpler example well covered and often used in existing analysis literature, including many demonstrations of uncertainty analysis. The PRISM model provides a more realistic but also more specialised example. The additional solving cost of the PRISM model allowed us to better consider how we apply the algorithms.

7.2 Limitations

The main limitation of the proposed solution is the computational cost in the face of uncertainty. In Section 2.5 we discussed the many uncertain aspects in the problem domain. These make it difficult to produce accurate algorithm results at a low computational cost. The parameter uncertainty modelling involves exploring the effect of relatively small changes in the parameter values (which themselves may contribute to random inputs). Detecting the effect of these small changes can require many model experiments. Uncertainty and sensitivity analysis methods tend to avoid this requirement since it does not scale well with an increasing number of inputs or model cost. Since the proposed algorithms do

the equivalent of multiple uncertainty analysis runs, it is obviously even more important.

We proposed many ways to improve efficiency in an attempt to overcome this limitation but for some costly models or models with many important parameters the methods will not be practical without a long run time or significant processing power. Where possible we have enabled parallel execution, working towards the assumption that distributed computing may be required in certain situations. One could also argue that if a company is interested in the result of a very complex model, they also more likely to be able to invest the necessary computing resources for a costly analysis.

7.3 Future Work

This section describes possible directions for future work in the problem area. We consider four classifications of future work: Additional or more complex problem constraints (Chapter 3), Parameter uncertainty modelling (Chapter 4), Alternative solving algorithms (Chapter 5), and Additional Results validation (Chapter 6) .

7.3.1 Problem Constraints

In Chapter 3 we described the data collection problem with objective functions and constraints. These constraints restricted the set of valid strategies. Data collection cost and budget were an important part of the constraints. In practice one is also restricted by the available data sources and we allowed configuration of the sample size minimum value, maximum value, and increments for a data source.

These constraints could be made more complex or additional constraints added to better represent a data collection scenario. In our implementation we had linear cost functions based upon sample size but for many collection methods you would want something more complex, such as a reducing sample cost as sam-

ple size increases. Considering additional constraints, one could separate time from the current abstraction. The time available for collection at a per data source level would change the maximum sample size modelling, and constrain whole strategies. We currently allow any combination of data sources within a strategy (as long as other constraints are met) but it is plausible that the use of one data source makes another unavailable, for example if doing one collection method could affect and invalidate the data of another. The only way right now to accommodate many of these would be by manually configuring each strategy to test, so assisting and automating the constraint specification could be explored further.

7.3.2 Parameter Uncertainty Modelling

This section discusses future work in generating input parameter values and designing experiments with these values.

One of the most important parts of solving the data collection optimisation problem is estimating and modelling a strategy's parameter uncertainty. This representation is fed into the model to analyse each strategy. How we model the current parameter data uncertainty and then make predictions on future collection are a fundamental part of all solving algorithms. We discussed three approaches in Chapter 4 but the overall framework was designed in such a way that would allow alternatives to be added and evaluated.

These methods generate sets of experiment values for each input parameter. Each input parameter's data uncertainty is modelled and a value generated independently. Looking at the existing approach there is also the opportunity for further exploiting of stratification and design of experiments. The current design of experiments is either none or all combinations. The implemented solution offers basic random stream like usage, where generated values are used and paired in order they are generated. Alternatively, it can create experiments using all combinations of the generated experiment values. The latter is the best solution for covering the input space but can become costly with many input parameters. Efficiently designing experiments that cover the input space is a well researched

problem. Incorporating and evaluating other techniques not covered in the thesis would be beneficial.

The parameter uncertainty modelling looks at a small number of measurable factors when making estimations. The most important of which are the variance and the number samples. In Section 3.2 we discussed other data qualities that may be important when evaluating data and data sources. Many of these are difficult to measure, especially in a way which is comparable across data sources, and this is an interesting problem by itself. Beyond that you can consider whether these metrics could be accommodated in our approach of testing strategy qualities through the model. For example a simple but difficult to calibrate approach would be to include multipliers altering the other algorithm inputs or results based upon other the metrics.

7.3.3 Alternative Solving Algorithms

Chapter 5 described optimisation solving algorithms for finding the optimal strategy or strategies. Three of these either tested all strategies or incrementally improved a strategy. With many input parameters, many data sources or very flexible data source options there may be a lot of valid strategies to test. In Section 5.6 we discussed other ways of reducing the problem or searching the space but these were not all fully formalised or implemented. Based upon these ideas and existing optimisation techniques, other solving algorithms could be devised to efficiently explore and solve a large strategy space.

As part of the Iterative Algorithm (Chapter 5) we also explored the idea of separating parameter uncertainty from simulation uncertainty with the use of ANOVA. The idea being that a strategy needs to be good enough that parameter uncertainty is no longer noticeable, and only simulation uncertainty remains. While the results were inconclusive, similar techniques based upon the same idea definitely warrant further research.

Parallel execution options were included in the MATLAB implementation to speed up strategy and experiment evaluation. These could be expanded further to multiple machines using MATLAB Distributed Computing Server [58]. For

very complex models or solution spaces it would appropriate and interesting to extend the solving to use cloud computing [57, 93] resources, such as Amazon EC2 [94] which has recently been added to MATLAB Distributed Computing Server.

7.3.4 Additional Results Validation

In Chapter 6 we demonstrated the method with two different models over multiple examples. Like all new methods, the solutions described in this thesis will need further testing both using additional models and by being incorporated into the modelling lifecycle. Other types of models could be used, as well as even more complex models with a long simulation time or many inputs. These would further test the computational requirements of running the solving algorithms.

To further evaluate the solving algorithms, they need to be used with a real active case study. The optimal data collection strategies would then be implemented and real data collected. This alone would allow you to compare the resulting data against the predicted improvements, testing and then tweaking the parameter uncertainty modelling techniques. If possible a control strategy would be implemented, created by expert opinion or using other methods. This kind of case study based evaluation was planned but fell through for two different studies. Long term validation of modelling and predictions is an important aspect for their usage but it is difficult to do in practice, especially with redundancy for comparison. For security, part of the problem is the rapidly changing systems being modelled and the surrounding landscape.

The ideal, but most costly way to evaluate these algorithms would be to use a well understood model and then do a lot of data collection for all possible data sources. This collection would be well beyond that recommended by the strategies, ideally the maximum collection allowed for each data source. One could then compare all strategy recommendations against strategies created after the fact, based upon the data collected. It would be the perfect test and allow major calibration of the methods but it would obviously be very costly.

References

[1] M. Freimer and L. Schruben, "Collecting data and estimating parameters for input distributions," in *Proceedings of the 34th Winter Simulation Conference*, 2002, pp. 392–399.

[2] R. Cain and A. Van Moorsel, "Optimization of Data Collection Strategies for Model-Based Evaluation and Decision-Making," in *IEEE/IFIP 42nd International Conference on Dependable Systems & Networks (DSN)*, Jun. 2012.

[3] P. Limbourg, "Dependability Modelling under Uncertainty: An Imprecise Probabilistic Approach," *Studies in Computional Intelligence*, vol. 148, Sep. 2008.

[4] J. Fitzgerald, P. Larsen, K. Pierce, M. Verhoef, and S. Wolff, "Collaborative modelling and co-simulation in the development of dependable embedded systems," English, in *Integrated Formal Methods*, ser. Lecture Notes in Computer Science, vol. 6396, Springer Berlin Heidelberg, 2010, pp. 12–26, ISBN: 978-3-642-16264-0. DOI: http://dx.doi.org/10.1007/978-3-642-16265-7_2.

[5] B. R. Haverkort, *Performance of computer communication systems: a model-based approach*. J. Wiley, 1998.

[6] R. R. Barton, "Tutorial: Input uncertainty in output analysis," in *Proceedings of the 2012 Winter Simulation Conference*, IEEE, 2012, pp. 1–12.

[7] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 37th Winter Simulation Conference*, 2005, p. 143.

[8] L. F. Konikow and J. D. Bredehoeft, "Ground-water models cannot be validated," *Advances in Water Resources*, vol. 15, no. 1, pp. 75–83, 1992.

[9] N. Oreskes, K. Shrader-Frechette, K. Belitz, *et al.*, "Verification, validation, and confirmation of numerical models in the earth sciences," *Science*, vol. 263, no. 5147, pp. 641–646, 1994.

[10] J. Cariboni, D. Gatelli, R. Liska, and A. Saltelli, "The role of sensitivity analysis in ecological modelling," *Ecological Modelling*, vol. 203, no. 1-2, pp. 167–182, Apr. 2007.

[11] J. P. C. Kleijnen, "Factor Screening in Simulation Experiments: Review of Sequential Bifurcation," in *Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman*, Springer-Verlag, 2009, pp. 1–15.

[12] T. Homma and A. Saltelli, "Importance measures in global sensitivity analysis of nonlinear models," *Reliability Engineering & System Safety*, vol. 52, no. 1, pp. 1–17, 1996.

[13] K. Chan, A. Saltelli, and S. Tarantola, "Sensitivity analysis of model output: variance-based methods make the difference," in *Proceedings of the 29th Winter Simulation Conference*, 1997, pp. 261–268.

[14] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, and J. Cariboni, *Global sensitivity analysis: the primer*. Wiley, 2008.

[15] J. Hamel, M. Li, and S. Azarm, "Design Improvement by Sensitivity Analysis Under Interval Uncertainty Using Multi-Objective Optimization," *Journal of mechanical design*, vol. 132, 2010.

[16] A. Saltelli, M. Ratto, S. Tarantola, and F. Campolongo, "Sensitivity analysis practices: Strategies for model-based inference," *Reliability Engineering & System Safety*, vol. 91, no. 10-11, pp. 1109–1125, 2006.

[17] A. Saltelli, "Global sensitivity analysis: an introduction," in *Proceedings of 4th International Conference on Sensitivity Analysis of Model Output (SAMO'04)*, 2004, 27–43.

[18] E. Plischke, E. Borgonovo, and C. L. Smith, "Global sensitivity measures from given data," *European Journal of Operational Research*, vol. 226, no. 3, pp. 536–550, 2013, ISSN: 0377-2217. DOI: http://dx.doi.org/10.1016/j.ejor.2012.11.047. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712008995.

[19] I. M. Sobol', "Sensitivity estimation for nonlinear mathematical models," *Matematicheskoe Modelirovanie*, vol. 2, no. 1, pp. 112–118, 1990, MMCE (1993) (in English).

[20] D. M. Hamby, "A comparison of sensitivity analysis techniques," *Health Physics*, vol. 68, no. 2, pp. 195–204, 1995.

[21] B. Sudret, "Global sensitivity analysis using polynomial chaos expansions," *Reliability Engineering & System Safety*, vol. 93, no. 7, pp. 964–979, 2008, Bayesian Networks in Dependability, ISSN: 0951-8320. DOI: http://dx.doi.org/10.1016/j.ress.2007.04.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832007001329.

[22] A. Saltelli and P. Annoni, "How to avoid a perfunctory sensitivity analysis," *Environmental Modelling and Software*, vol. 25, no. 12, pp. 1508–1517, 2010, cited By 134.

[23] F. Campolongo, A. Saltelli, and J. Cariboni, "From screening to quantitative sensitivity analysis. A unified approach," *Computer Physics Communications*, vol. 182, no. 4, pp. 978–988, Apr. 2011.

[24] H. Christopher Frey and S. R. Patil, "Identification and review of sensitivity analysis methods," *Risk analysis*, vol. 22, no. 3, pp. 553–578, 2002.

[25] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[26] I. Sobol' and S. Kucherenko, "Derivative based global sensitivity measures and their link with global sensitivity indices," *Mathematics and Computers in Simulation*, vol. 79, no. 10, pp. 3009–3017, 2009, ISSN: 0378-4754. DOI: http://dx.doi.org/10.1016/j.matcom.2009.01.023. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378475409000354.

[27] ——, "A new derivative based importance criterion for groups of variables and its link with the global sensitivity indices," *Computer Physics Communications*, vol. 181, no. 7, pp. 1212–1217, 2010, ISSN: 0010-4655. DOI: http://dx.doi.org/10.1016/j.cpc.2010.03.006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S001046551000086X.

[28] M. McKay, J. Morrison, and S. Upton, "Evaluating prediction uncertainty in simulation models," *Computer Physics Communications*, vol. 117, no. 1-2, pp. 44–51, 1999.

[29] A. Bar Massada and Y. Carmel, "Incorporating output variance in local sensitivity analysis for stochastic models," *Ecological Modelling*, vol. 213, no. 3-4, pp. 463–467, 2008.

[30] J. Ascough, T. Green, L. Ma, and L. Ahuja, "Key criteria and selection of sensitivity analysis methods applied to natural resource models," *International Congress on Modeling and Simulation Proceedings*, 2005.

[31] J. P. C. Kleijnen, "Searching for important factors in simulation models with many factors: Sequential bifurcation," *European Journal of Operational Research*, no. 96, pp. 180–194, Dec. 1996.

[32] W. Shi and Z. Liu, "Factor screening for simulation with multiple responses: Sequential bifurcation," *European Journal of Operational Research*, 2014.

[33] X. Wang, Y. Tang, and Y. Zhang, "Orthogonal arrays for estimating global sensitivity indices of non-parametric models based on ANOVA high dimensional model representation," *Journal of Statistical Planning and Inference*, vol. 142, no. 7, pp. 1801–1810, 2012, ISSN: 0378-3758. DOI: http://dx.doi.org/10.1016/j.jspi.2012.02.043. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378375812000900.

[34] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[35] C. J. Roy and W. L. Oberkampf, "A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 25-28, pp. 2131–2144, Jun. 2011.

[36] J. C. Helton, J. Johnson, C. Sallaberry, and C. Storlie, "Survey of sampling-based methods for uncertainty and sensitivity analysis," *Reliability Engineering & System Safety*, vol. 91, no. 10-11, pp. 1175–1209, 2006.

[37] W. L. Oberkampf, J. C. Helton, C. A. Joslyn, S. F. Wojtkiewicz, and S. Ferson, "Challenge problems: uncertainty in system response given uncertain parameters," *Reliability Engineering & System Safety*, vol. 85, no. 1-3, pp. 11–19, Jul. 2004.

[38] S. G. Henderson, "Input modeling: input model uncertainty: why do we care and what should we do about it?" In *Proceedings of the 2003 Winter Simulation Conference*, Winter Simulation Conference, Dec. 2003, pp. 90–100.

[39] F. Zouaoui and J. Wilson, "Accounting for input model and parameter uncertainty in simulation," in *Proceedings of the 33nd Winter Simulation Conference*, 2001, pp. 290–299.

[40] A. Law and W. Kelton, *Simulation modeling and analysis*, ser. McGraw-Hill series in industrial engineering and management science. McGraw-Hill, 2000, ISBN: 9780070592926. [Online]. Available: http://books.google.co.uk/books?id=QqkZAQAAIAAJ.

[41] R. C. H. Cheng, "Selecting input models," in *Proceedings of the 1994 Winter Simulation Conference*, IEEE, 1994, pp. 184–191.

[42] S. Sankararaman and S. Mahadevan, "Distribution type uncertainty due to sparse and imprecise data," *Mechanical Systems and Signal Processing*, vol. 37, no. 1-2, pp. 182–198, 2013, ISSN: 0888-3270. DOI: http://dx.doi.org/10.1016/j.ymssp.2012.07.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327012002713.

[43] S. E. Chick, "Input distribution selection for simulation experiments: accounting for input uncertainty," *Operations Research*, vol. 49, no. 5, pp. 744–758, 2001.

[44] J. P. C. Kleijnen, "Sensitivity analysis versus uncertainty analysis: when to use what?" In *Predictability and Nonlinear Modelling in Natural Sciences and Economics*, J. Grasman and G. van Straten, Eds., Springer Netherlands, 1994, pp. 322–333, ISBN: 978-94-010-4416-5. DOI: http://dx.doi.org/10.1007/978-94-011-0962-8_27.

[45] R. R. Barton, R. C. H. Cheng, S. Chick, S. Henderson, A. M. Law, L. Leemis, B. Schmeiser, L. Schruben, and J. Wilson, "Panel on current issues in simulation input modeling," in *Proceedings of the Winter Simulation Conference 2002*, 2002, pp. 353–369.

[46] R. R. Barton and B. L. Nelson, "Quantifying input uncertainty via simulation confidence intervals," *INFORMS Journal on Computing*, vol. 26, no. 1, pp. 74–87, 2014.

[47] R. C. H. Cheng and W. Holland, "Calculation of confidence intervals for simulation output," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 14, no. 4, pp. 344–362, 2004.

[48] J. P. C. Kleijnen, "Risk analysis and sensitivity analysis: antithesis or synthesis?" *ACM SIGSIM Simulation Digest*, vol. 14, no. 1-4, pp. 64–72, 1983.

[49] S. Ng and S. Chick, "Reducing parameter uncertainty for stochastic systems," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 16, no. 1, pp. 26–51, 2006.

[50] R. R. Barton and L. Schruben, "Uniform and Bootstrap Resampling of Empirical Distributions," in *Proceedings of the 25th Winter Simulation Conference*, New York, NY, USA: ACM, 1993, pp. 503–508.

[51] ——, "Resampling methods for input modeling," in *Proceedings of the 2001 Winter Simulation Conference*, vol. 1, 2001, pp. 372–378.

[52] R. R. Barton, *Presenting a more complete characterization of uncertainty: Can it be done?* 2007.

[53] B. E. Ankenman, B. L. Nelson, and J. Staum, "Stochastic kriging for simulation metamodeling," in *Proceedings of the 2008 Winter Simulation Conference*, 2008, pp. 362–370.

[54] B. E. Ankenman and B. L. Nelson, "A quick assessment of input uncertainty," in *Proceedings of the 2012 Winter Simulation Conference*, IEEE, 2012, pp. 1–10.

[55] E. Song and B. L. Nelson, "A quicker assessment of input uncertainty," in *Proceedings of the 2013 Winter Simulation Conference*, IEEE, 2013, pp. 474–485.

[56] F. Pappenberger and K. J. Beven, "Ignorance is bliss: or seven reasons not to use uncertainty analysis," *Water Resources Research*, vol. 42, no. 5, 2006, W05302, ISSN: 1944-7973. DOI: http://dx.doi.org/10.1029/2005WR004820.

[57] Y. Liu, A. Y. Sun, K. Nelson, and W. E. Hipke, "Cloud computing for integrated stochastic groundwater uncertainty analysis," *International Journal of Digital Earth*, vol. 6, no. 4, pp. 313–337, Jul. 2013.

[58] MathWorks, *MATLAB Parallel Computing Toolbox*, http://uk.mathworks.com/products/parallel-computing/, Accessed 2015-01-15, 2014.

[59] A. D. Kiureghian and O. Ditlevsen, "Aleatory or epistemic? Does it matter?" *Structural Safety*, 2009.

[60] S. Ng and S. Chick, "Reducing input parameter uncertainty for simulations," in *Proceedings of the 33rd Winter Simulation Conference*, 2001, pp. 364–371.

[61] S. E. Chick, "Bayesian methods for simulation," in *Proceedings of the 2000 Winter Simulation Conference*, IEEE, 2000, pp. 109–118.

[62] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: a tutorial," *Statistical Science*, 1999.

[63] C. B. Storlie, L. P. Swiler, J. C. Helton, and C. J. Sallaberry, "Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models," *Reliability Engineering & System Safety*, vol. 94, no. 11, pp. 1735–1763, Jul. 2009.

[64] P. Närman, P. Johnson, R. Lagerström, U. Franke, and M. Ekstedt, "Data Collection Prioritization for System Quality Analysis," *Electronic Notes in Theoretical Computer Science*, vol. 233, pp. 29–42, 2009.

[65] A. Skoogh, J. Michaloski, and N. Bengtsson, "Towards continuously updated simulation models: combining automated raw data collection and automated data processing," in *Proceedings of the 2010 Winter Simulation Conference*, 2010, pp. 1678–1689.

[66] J. P. C. Kleijnen, "Design of experiments: overview," *Proceedings of the 40th Winter Simulation Conference*, pp. 479–488, 2008.

[67] P. W. Glynn and D. L. Iglehart, "Importance sampling for stochastic simulations," *Management Science*, vol. 35, no. 11, pp. 1367–1392, 1989.

[68] A. Owen and Y. Zhou, "Safe and Effective Importance Sampling," *Journal of the American Statistical Association*, vol. 95, pp. 135–143, 2000.

[69] J. P. C. Kleijnen and R. G. Sargent, "A methodology for fitting and validating metamodels in simulation," *European Journal of Operational Research*, vol. 120, no. 1, pp. 14–29, 2000.

[70] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-based evaluation: from dependability to security," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 48–65, 2004.

[71] A. Beautement, R. Coles, J. Griffin, C. Ioannidis, B. Monahan, D. Pym, M. A. Sasse, and M. Wonham, "Modelling the human and technological costs and benefits of USB memory stick security," *Managing Information Risk and the Economics of Security*, pp. 141–163, 2009.

[72] S. Arnell, A. Beautement, P. Inglesant, B. Monahan, D. Pym, and M. A. Sasse, "Systematic Decision Making in Security Management Modelling Password Usage and Support," *HP Tech Report*, pp. 1–32, Mar. 2011.

[73] S. Parkin, R. Yassin Kassab, and A. Van Moorsel, "The impact of unavailability on the effectiveness of enterprise information security technologies," *Service Availability*, pp. 43–58, 2008.

[74] W. Zeng and A. Van Moorsel, "Quantitative Evaluation of Enterprise DRM Technology," *Electronic Notes in Theoretical Computer Science*, vol. 275, pp. 159–174, 2011.

[75] R. C. Thomas, M. Antkiewicz, P. Florer, S. Widup, and M. Woodyard, "How Bad Is It? – A Branching Activity Model to Estimate the Impact of Information Security Breaches," *12th Workshop on the Economics of Information Security (WEIS)*, 2013. DOI: `http://dx.doi.org/10.2139/ssrn.2233075`.

[76] R. Anderson and T. Moore, "Information Security Economics - and Beyond," in *DEON '08: Proceedings of the 9th international conference on Deontic Logic in Computer Science*, Berlin, Heidelberg: Springer-Verlag, Jul. 2008, pp. 49–49.

[77] S. Parkin, A. Van Moorsel, P. Inglesant, and M. A. Sasse, "A stealth approach to usable security: helping IT security managers to identify workable security solutions," in *Proceedings of the 2010 workshop on New security paradigms (NSPW)*, ACM Request Permissions, Sep. 2010, pp. 33–50.

[78] MathWorks, *MATLAB R2013a–R2014b*, `http://www.mathworks.co.uk`, Accessed 2014-08-01.

[79] ——, *SimEvents*, `http://uk.mathworks.com/products/simevents/`, Accessed 2015-01-15.

[80] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, G. Gopalakrishnan and S. Qadeer, Eds., ser. LNCS, vol. 6806, Springer, 2011, pp. 585–591.

[81] ——, *PRISM - Probabilistic Symbolic Model Checker 4.2*, `http://www.prismmodelchecker.org`, Accessed 2014-08-01, 2014.

[82] R. H. Frank and B. Bernanke, *Principles of economics*. McGraw-Hill, 2004.

[83] C. Batini and M. Scannapieca, *Data Quality*, ser. Concepts, Methodologies and Techniques. Springer, Sep. 2006.

[84] R. Wang and D. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of management information systems*, vol. 12, no. 4, pp. 5–33, 1996.

[85] Y.-C. Ho, "Introduction to special issue on dynamics of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 3–6, Jan. 1989.

[86] A. Van Moorsel, L. Kant, and W. H. Sanders, "Computation of the asymptotic bias and variance for simulation of Markov reward models," in *Proceedings of the 29th Annual Simulation Symposium*, IEEE, 1996, pp. 173–182.

[87] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. John Wiley & Sons: John Wiley & Sons, 2010.

[88] B. Efron and R. Tibshirani, "Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy," *Statistical science*, pp. 54–75, 1986.

[89] B. M. Ayyub and R. H. McCuen, *Probability, Statistics, and Reliability for Engineers and Scientists*, 3rd. CRC press, 2011.

[90] H. Levene, "Robusts tests for the equality of variance," in *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, Stanford University Press, 1960.

[91] A. Saltelli and T. Homma, "Sensitivity analysis for model output: Performance of black box techniques on three internation benchmark exercises," vol. 13, no. 1, pp. 73–94, Nov. 2001.

[92] J. C. Mace, C. Morisset, and A. P. A. van Moorsel, "Quantitative workflow resiliency," in *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I*, 2014, pp. 344–361. DOI: http://dx.doi.org/10.1007/978-3-319-11203-9_20.

[93] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672.

[94] *Amazon Elastic Compute Cloud (Amazon EC2)*, http://aws.amazon.com/ec2/, Accessed 2015-01-15, Amazon Web Services Inc., 2014.

[95] Oracle, *Java SE Development Kit 7*, http://www.oracle.com/technetwork/java/javase/overview/index.html, Accessed 2014-08-01.

Appendices

APPENDIX A

Implementation Details

Chapter 5 explained different solving algorithms for the data collection optimisation problems described in Chapter 3, including Basic Exhaustive Algorithm, Basic with Importance Sampling Algorithm, and the Iterative Algorithm. These algorithms and the supporting methods from previous chapters were implemented in MATLAB. This appendix provides a structural overview and functional explanation of the MATLAB tool implementation and supporting software used in the examples. It explains how the key components work and interact. This enables one to understand the software for execution or extension, and bridge the method explanations to the results in Chapter 6.

The rest of the appendix is structured as follows: Section A.1 provides a high level overview of the modules, key functions and classes, and how they interact. Section A.2 details they main data classes and any supporting data structures. Section A.3 explains the MATLAB functions, including the input modelling functions in Section A.3.1 and algorithm evaluation functions in Section A.3.2.
Section A.4 discusses running tests and examples, as well as automated repeated execution. Section A.5 summarises the content.

A.1 Structure

This section provides an overview of the implementation structure and explains the main components. The implementation was designed to be generic, to be eas-

ily usable with different models and scenarios, and to be modular, to allow interchanging and testing of different methods. The implementation is largely written in MATLAB but also uses Java [95] and PRISM for the execution of models. It is split into: *MATLAB Core*, *MATLAB MM1*, and *PRISM* modules. MATLAB Core is where most of the work is done with all of the generic classes and functions applicable to all models and examples. MATLAB MM1 provides functions and scripts for conducting MM1 queue based examples. The PRISM module enables integration of external model execution using PRISM model checker and includes related scripts for those examples.

Three primary classes store and manage the information about inputs (*ModelInput*), data sources (*ModelDataSource*), and strategies and their results (*ModelDataStrategy*). These classes are used to create strategies and analyse strategies using the algorithms described in Chapter 3.

Each Algorithm from Chapter 5 has its own function to complete the process. The functions are largely based around different ways of executing and evaluating a shared single strategy evaluation function, *analyseStrategy*. The algorithm functions take a variety of options to allow customisation and are not specific to one parameter uncertainty modelling method or model. These can both be defined separately.

The parameter uncertainty modelling methods, explained in Chapter 4 are provided by ModelInput and ModelDataSource, and with a small amount of effort can be used interchangeably.

Models are wrapped and represented by a MATLAB function, which analytically solves the model, simulates the model to the desired accuracy, or interacts with external simulation software as required. As long as the model can be represented and interfaced in the required form, it can be used with any algorithm.

Examples are created and executed using scripts that use the above to setup the data collection scenario, define the model function, and then call the chosen optimisation algorithm function. Additional automation functions allow for algorithms to be repeatedly executed with different parameters to allow for comparisons to be made. The following sections describe key data classes and an

explanation of the major functions.

A.2 Data Classes

The three primary data classes hold and control the data needed to setup and execute each of the algorithms. This section describes those classes and how they relate to each other and the larger functions.

A.2.1 ModelInput

ModelInput class represents an input parameter p_i which is used in the model and available for additional data collection. It is initialised with a numeric identifier, name and data type (*ModelDataType*). The data type allows transformations or restrictions on the input value used, such as rounding to integer or transforming it into a rate. Using the other data classes described below, *ModelInput* is linked to a list of the available data sources, and a chosen active source (or sources) and sample size for the current strategy. The class provides functions to use these one or more data sources to produce values for model experiments that represent the parameter uncertainty. This includes the methods discussed earlier: CLT normal distribution sampling, interval-based CLT normal distribution sampling, and bootstrap resampling. Each has configurable options and are more fully explained in Section A.3.1.

A.2.2 ModelDataSource

Each data source D_j and any existing data are represented using *ModelDataSource* and *DCOExistingData* classes. It is assumed that some existing data has been collected, and that DCOExistingData holds the $M(D_j)$ existing samples or (if unavailable) just a sample mean, sample variance, and sample size for the data. *ModelDataSource* is a wrapper for this existing data and properties of the planned collection of strategy. *ModelDataSource* is initialised with an identifier, a mapping of the source to input parameters (similar to $d_{i,j}$ of Section 3.3.2), existing data for

one or more input parameters to be populated, and a sample cost. Additional properties can be set such as the sample increment η_j, used when increasing sample sizes, the minimum $N_{min}(D_j)$ and maximum $N_{max}(D_j)$ allowed samples sizes, and the start up cost v_j for the collection. The class provides helper methods for generating input values in *ModelInput*.

A.2.2.1 Cost Functions

The total cost of strategy $T(s)$ is calculated by computing the collection cost for each data source in the strategy. Each data source has a cost function $C_j(N_j)$ (as described in Section 3.3.2). In the *ModelDataSource* class, the cost function can be set as a property and uses a provided linear sample cost function by default. This calculates the source's total cost by

$$C_j(N_j) = v_j + (N_j \times c_j)$$

Where v_j is start up cost and c_j is the cost per sample. The start up cost is the initial outlay required to collect the first set of samples with the associated source. This is just one example of non-uniform total cost. The cost function can easily be replaced and configured as appropriate. A function for computing the maximum samples within budget must also be provided. This function effectively solves the cost function for the number of samples given a maximum total cost and it is required for generating strategies (see Section A.2.3).

A.2.3 ModelDataStrategy

A data collection strategy is a combination of data sources and $N(D_j)$ a number of samples to collect for each source, which links to a set of input parameters of a model. *ModelDataStrategy* holds a list of the input parameters (*ModelInput*), information about available and active data sources (*ModelDataSource*), and all the related model input and output values created when evaluating that strategy. *ModelDataStrategy* provides functions to:

- Connect the *ModelInput* and *ModelDataSource* instances based on valid combinations and the current strategy's configuration.

- Calculate the mean and ouput variance over model results for the strategy, both via the standard method and by using importance sampling weightings. It also computes the variance of the variance result for assessing the algorithm accuracy.

- Calculate the total cost collection for the strategy when using the current data source configuration

- Complete an ANOVA using the random effects model for the strategy's results to estimate main and interaction effects of the input parameters.

Strategies using *ModelDataStrategy* can be set up manually or created by using the *generateStrategies* function. This dedicated function takes an array of model inputs and array of data sources. It produces all valid strategies within a given budget. It is achieved by first translating all valid input-data source pairings into a set of base strategies, which feature one data source for each input. Each data source has a sample size increment property. For each base strategy, those sample increments, the data source cost functions, the available budget, and any sample size limits are used to exhaustively create all combinations of sample allocations within budget. All input parameters in the generated strategies have one primary data source, but a parameter may receive additional data from other data sources when appropriate (see Section 3.3.2.2).

With many sources per input, small data source sample increments, or a large budget, the generateStrategies function can result in many strategies to evaluate. In some situations one may need to start with larger sample increments to explore the space and then more flexible options to fine tune the solution. The function also offers the option to limit the generated strategies to those with a total cost between a chosen value and a maximum value. This option is beneficial when the optimal strategies are expected to be those that spend close to the collection budget, or generate batches in total cost ranges.

A.3 Main Functions & Execution

A.3.1 Input Functions

Chapter 4 explained a number of ways of modelling input parameter uncertainty as model experiments. The MATLAB implementation provides three of these as experiment generators: Central Limit Theorem (CLT) based Normal distribution sampling, CLT-based Normal distribution sampling with intervals, and bootstrap resampling. Generically all of these operate at a per input parameter level and take a function parameter for the number of values required. The functions use the input's associated data source information to generate the experiment values. Since all these generator functions follow the same interface they can be used interchangeably with the Optimisation Solving Algorithms.

An input parameter can have one or more active data sources. If the parameter is populated using data from multiple active data sources generating experiment values uses a data combining mode. This is either *All*, which uses or attempts to combine information from all active sources into the parameter uncertainty modelling, or *Best Estimated* where only the predicated best source is used. The effects of these modes and the defaults are described with each function (for more extended explanation see Section 4.7).

generateValuesNormalSampling Using the method described in Section 4.3, a normal distribution is constructed based upon the existing data from the data source and the total number of samples to be collected. The function randomly samples this distribution to produce values for the associated input parameter. Additional transformations may be applied depending upon the parameter type such as rounding to integer or converting to a rate. *Best Estimated* uses only the data source with the smallest estimated standard deviation when taking into account strategy sample sizes. *All* mode: creates a standard deviation estimate by weighting the standard deviation estimate contribution from each active source. The weight is the ratio of the data source samples to the total samples for an input (across all active sources).

generateValuesNormalIntervalSampling This generation function is similar to the above normal distribution sampling for parameter uncertain modelling, but uses interval based sampling (Section 4.4). Rather than randomly sampling from the entire normal distribution, it is split into a number of intervals with equal probability and an equal number of distribution samples are allocated to each interval. The sampling within intervals can be either random or at fixed distances, such as the mid-points of each interval. This technique is less random but allows guaranteed good coverage of the normal distribution, and therefore the modelled uncertainty, even when using a smaller number of experiment values.

generateValuesBootstrap The Bootstrap resampling technique is explained in Section 4.6 including how it can be applied to this context. Given an array of existing samples, an experimental value is created by repeatedly resampling with replacement from the existing samples and then taking the mean of the resulting 'bootstrap samples'. In this case the number of times to resample depends upon the total number of collection samples allocated to the source $M(D_j) + N(D_j)$ (existing + planned). This results in one experiment value for the input parameter, the process is repeated for the required number of values. The preferred combining mode for this function is *All* which conducts the bootstrap resampling over all actives sources for an input. If there is more than one active data source, there will be multiple bootstrapped sample sets. An experiment value for the input parameter is created by simply taking for the mean over all the bootstrapped samples. *Best Estimated* mode uses only the active source with the smallest sample variance.

A.3.1.1 Parameter Validating

Parameter values are validated at an input parameter and strategy level. Each *ModelInput* can include an optional minimum and maximum value. These can be set to any value or pre-populating based on the input parameter type e.g. a parameter of probability type can be restricted to between 0 and 1. The generated

parameter value is internally checked against this valid range when using the previous input functions. If it falls outside the range, a value will be automatically re-generated until the maximum number of regenerations is reached (this can be defined per parameter). Overflowing the regeneration limit maximum shows a problem with the data source configuration that needs to be addressed.

During strategy evaluation, each experiment (a complete set of values, one for each parameter) can also be validated. Each solving algorithm function can take an optional *parameter checking function*. This function can be created and customised for each model's requirements. The function must compare the parameter values in an experiment against the valid space of the model, and return a boolean for whether it passed. It is used by the *analyseStrategy* function. For failing experiments it will either: re-generate the experiment (for random experiments with no design), or remove the experiment (when using the 'all combinations' full factorial design). Like parameter regeneration, there is a configurable amount for the maximum number of experiment regenerations or deletions before an error is thrown.

A.3.2 Optimisation Solving Algorithm Functions

Specialised implementation was created three solving algorithms from Chapter 5. Each has its own function to complete the algorithm process, Basic Exhaustive Algorithm as *analyseStategies*, Basic Exhaustive Algorithm with Importance Sampling as *analyseStrategiesWithImportanceSampling*, and the ANOVA based Iterative Algorithm as *iterativeStrategyAlgorithm*. All use a shared *analyseStrategy* function to execute the evaluation of a single strategy. The *analyseStrategy* function uses the model to test the effect of the strategy's parameter uncertainty on the model output. It takes the following parameters:

Strategy the *ModelDataStrategy* which includes information about data sources for each input. It holds the generated experiments, the model and strategy results.

Input function a handle for the function to use to generate experiment values.

One of the input functions from Section A.3.1 can be used. It can also be any function that takes one parameter, the number of experiments, and returns the same number of values. An anonymous function may be used to wrap other functions enabling them to meet specification.

Model function the model we are using and planning data collection for. The model function is either called once per experiment with a single output, or in batch mode, once per batch of experiments returning an array of model outputs. The model function is usually a wrapper interfacing with the actual model functions in MATLAB or external modelling tools such as PRISM. In batch mode, the model function can offer parallelised execution.

Number of repetitions per experiment The number of times to repeat each experiment with the model. Repeated execution can be important when simulation variance is high and is necessary for ANOVA calculations.

Maximum number of experiments R^*: The number of experiments used to analyse the strategy $R \leq R^*$. R is calculated based upon whether the all combinations experimental design is used and the number of repetitions per experiment. Equation A.1a is when plain random ordering is used, Equation A.1b is when all combinations mode is used.

$$R = k \times r. \tag{A.1a}$$

$$R = k^{|P|} \times r. \tag{A.1b}$$

Additional options include: whether all combinations of parameter values are used in generated experiments and if the model function accepts batches of experiments.

The three solving algorithm functions take the same algorithm parameters and some function specific parameters. The solving functions then use *analyseStrategy* in different ways resulting in varied amounts of model computation.

analyseStrategies Completes the Basic Exhaustive Algorithm (Section 5.2) by simply evaluating the shared *analyseStrategy* function for every strategy.

The results can then be compared or further constrained before an optimal strategy is chosen using an objective function (Section A.3.5).
Algorithm complexity: proportional to the number of strategies.
Important additional parameters: the algorithm can execute strategies in parallel if appropriate. The algorithm can take an optional batch size and stopping function (see Section 5.6 & Section A.3.3).

analyseStrategiesWithImportanceSampling Completes the Basic Algorithm with importance sampling (Section 5.3). It uses *analyseStrategy* for a single 'anchor strategy', specified in the parameters, and then importance sampling is used to create results for the other strategies. The importance sampling is achieved by weighting each model result by comparing the probability distribution functions of the parameter uncertainty modelling, for each input in the anchor strategy against those in each of the target strategies.
Algorithm complexity: one strategy is evaluated with the model but additional results may be needed if anchor strategy choice is poor or alternative data source choices are significantly different. Complexity increases proportional to the number of strategies but at a much slower rate than *analyseStrategies*.
Important additional parameters: importance sampling anchor strategy index and anchor strategy results if pre-calculated. The algorithm can execute strategies in parallel if appropriate.

iterativeStrategyAlgorithm Completes the Iterative Algorithm (Section 5.5) which uses the *analyseStrategy* to assess a single strategy once per iteration and uses the results to decide the next iteration. ANOVA is used to decide if the current strategy has diminished parameter uncertainty variance enough. The algorithm takes a starting strategy instead of many strategies. A *nextStrategy* function creates subsequent strategies by incrementing sample sizes at the end of each iteration based on the rules in Section 5.5.1. Execution stops if an optimal strategy is found or the constraint is reached.
Algorithm complexity: Varies but it should be smaller than the others as it explores only a limited path through the solution space.

Important additional parameters: an optional constraint function can be included. This can be used to constrain strategy evaluation to those within a set budget.

A.3.3 Batched Solving Algorithms and Stopping Functions

The batch-based algorithms: *Cheapest-First Search* and *Most-Expensive First Search* can be executed by specialised configuration of *analyseStrategies* or *analyseStrategiesWithImportanceSampling*. The strategies must be pre-sorted by cost based on the needs of the algorithm using the available sorting function. The sorted strategies are then passed to an algorithm function. Both algorithm functions accept a *batch size* and a *stopping function*. The batch size is how many strategies are evaluated between each test of the stop condition using the stopping function. To meet the requirements described in Section 5.6, a stopping function was created for each algorithm:

stopVarianceReached for the Cheapest-First Search Algorithm: The stopping condition tests if any of the strategies evaluated so far met the variance constraint $Var[g(Y)|s] \leq V$. V must be specified as a parameter. Once an optimal strategy is found, the function does not allow the algorithm to stop until all strategies of equal $T(s)$ total cost are evaluated. This makes sure all optimal strategies are found, not just the first optimal strategy.

stopVarianceDiverging for the Most-Expensive-First Search Algorithm: Based on the batches of strategies evaluated so far and their results, the function makes a prediction on whether the most optimal strategy (or strategies) have been found. This involves making sure the extremes of strategy total cost $T(s)$ have been covered and if the variance results are diverging. In this case, diverging is defined as the lowest $Var[g(Y)|s]$ in the latest batch of strategies being 10% more than the current lowest $Var[g(Y)|s]$. The percentage threshold is configurable. We also allow the specification of a minimum $T(s)$ difference between the first and stopping strategy.

The stopping function-solving algorithm interfacing uses two parameters, the strategy results so far and the batch size, and returns a stop boolean. The above functions need or allow additional parameters which can be defined before execution using an anonymous function.

A.3.4 Parallel Execution

All of the solving algorithms except the Iterative Algorithm can (optionally) evaluate strategies in parallel using the MATLAB Parallel Computing Toolbox [58]. The implementation is primarily aimed at multi-core evaluation on the same computer. Depending upon the modelling software used it could also be evaluated via a Distributed Computing Server using a local cluster.

Separately as part of the MATLAB-Model interfacing experiments per strategy can also be executed in parallel. This is especially important if the modelling software only supports single-threaded solving and one experiment at a time. In most situations these two parallel options will not be used together. With simple models or small solution spaces parallel execution may not always reduce computation time.

A.3.5 Objective Functions and Constraints

In Section 3.3.2 we presented our data collection problem as Optimisation Problems with two different possible objective functions dictating the optimisation goal. These looked at the problem from two different perspectives: *minimise the variance* from the strategy's uncertainty (within budget) and *minimise cost* (within some acceptable level of variance), with other possible constraints strategies.

$$Min_{s \in S} \;\; Var[g(Y)|s]$$

$$Min_{s \in S} \;\; T(s) \quad T(s) = \sum_{j=1}^{|D|} C_j$$

Most of the constraints (such as the collection budget) were encapsulated in either strategy generation function or by manual strategy configuration. The objective functions were implemented as separate MATLAB functions in the solution

with a shared generic structure. Each takes an array of strategies (*ModelDataStrategy* which includes the results) and zero to many function parameters. They return one (or more) optimal strategies depending upon the results and objective parameters. This allows the objective functions to be used interchangeably or swapped out for an alternative objective function.

Minimise variance function: The function attempts to minimise the output variance when testing each strategy with its parameter uncertainty representation. The function looks at the all the strategies output variance results and returns the smallest. It is assumed all strategies passed to the function are valid, meeting the required constraints including the collection budget. In the unlikely event that more than one strategy meets the minimise variance objective, by having the same and lowest variance, multiple strategies can be returned. An optional secondary objective minimise cost can be enabled to choose between the strategies that equally meet the first objective.

Minimise cost function: The objective function in MATLAB also encapsulates the maximum variance constraint (see Section 3.3.2). The objective aims to find the strategy with smallest total collection cost while also checking the strategy's result meets the constraint. The variance value is provided as function parameter. If strategy configurations options and data source costs are granular, it is possible that multiple strategies may meet the constraint and have the same minimum total cost. As before, a secondary objective can be enabled, in this case to minimise variance over the strategies chosen by the first objective.

Either or both functions can be run as the final stage of the Basic optimisation Algorithms, to find the optimal strategies, or as part of stopping condition evaluation.

A.4 Tests & Automation

This section describes simple tests for each function and support functions for automated repeated algorithm execution.

Tests and **optimisation examples** are executed using MATLAB scripts. Each of the major functions has an accompanying test script. These test scripts demonstrate usage and run the function using a range of the different options available. Each optimisation example script has a similar form. First it initialises the related inputs and data sources before setting up or generating one or more strategies. These strategies are evaluated using an Optimisation Algorithm function and the results are then displayed.

A.4.1 AutoRunStrategyAlgorithm

This function enables automated repeated execution of a solving algorithm with or without varying a number of algorithm options. The function can repeatedly execute a same algorithm to compare its results between executions. It can also test a optimisation solving algorithm while varying the following algorithm options:

Number of experiment values k the number of different values that are generated as part of modelling the input parameter uncertainty..

Number of experiment repetitions r the number of times the simulation or solving of an experiment is repeated..

Simulation Time t the lengthy of time or number of steps in one model simulation.

An additional helper class *DCOAlgorithmResult* was created to hold algorithm parameters and results when running many algorithm executions in a single session. The automated repeated execution algorithm uses this to track and return results.

A.5 Summary

This appendix provided a detailed explanation of a MATLAB implementation of the optimisation solving algorithms from Chapter 5 and related supporting methods from Chapter 4. It described the data structures and function architecture which allows for interchanging of models and uncertainty modelling methods with little to no changes to existing implementation.

APPENDIX B

PRISM Model Details

The following appendix provides additional details of the workflow represented as a PRISM model in Section 6.2.

B.1 Workflow Details

```
10 Tasks, numbered 1 to 10
5 Users, numbered 1 to 5
10 Permission Restrictions
5 Separation of Duties
1 Binding of Duties

Permission Restrictions
perm1 task 1 only user 3 and 5 have the necessary permissions
perm2 task 2 only user 1 and 5 have the necessary permissions
perm3 task 3 only user 2 have the necessary permissions
perm4 task 4 only user 1, 3, 4, and 5 have the necessary permissions
perm5 task 5 only user 1 and 2 have the necessary permissions
perm6 task 6 only user 2, 3, 4, and 5 have the necessary permissions
perm7 task 7 only user 2 and 3 have the necessary permissions
perm8 task 8 only user 3 has the necessary permissions
perm9 task 9 only user 2, 3, 4, and 5 have the necessary permissions
perm10 task 10 only user 3 and 4 have the necessary permission
```

```
Separation of Duties
sod1 task 6 and 8 cannot be completed by the same user
sod2 task 1 and 3 cannot be completed by the same user
sod3 task 3 and 5 cannot be completed by the same user
sod4 task 4 and 6 cannot be completed by the same user
sod5 task 1 and 2 cannot be completed by the same user

Binding of Duties
bod1 task 1 and 7 must be completed by the same user
```